ProVerif with Lemmas, Induction, Fast Subsumption, and Much More

Anonymized for submission to Security and Privacy 2021

August 10, 2021

Abstract

This paper presents a major overhaul of one the most widely used symbolic security protocol verifiers, ProVerif. We provide two main contributions. First, we extend ProVerif with lemmas, axioms, proofs by induction, natural numbers, and temporal queries. These features not only extend the scope of ProVerif, but can also be used to improve its precision (that is, avoid false attacks) and make it terminate more often. Second, we rework and optimize many of the algorithms used in ProVerif (generation of clauses, resolution, subsumption, ...), resulting in impressive speed-ups on large examples.

1 Introduction

Security protocols aim at securing communications. They are used in various applications: establishment of secure channels over the Internet, secure messaging, electronic voting, mobile communications, etc. Their design is known to be error prone and flaws are difficult to fix once a protocol is largely deployed. Hence a common practice is to analyze the security of a protocol using formal techniques and in particular automatic tools. For example, TLS 1.3 has been designed while research groups were developing formal models in parallel and suggesting modifications [BBK17].

Several tools have been proposed for automatized security analysis of protocols. Some tools focus at restricted classes of protocols for which the analysis is deemed to terminate. They typically focus on a bounded number of sessions like Avispa [ABB+05], DeepSec [CKR18], or Akiss [CCK12]. These tools are efficient at finding attacks on small protocols but quickly face state explosion for complex protocols. Hence for large and complex protocols, tools like Tamarin [SMCB12] and ProVerif [Bla14] are often preferred. They both offer a flexible framework to model a protocol and its primitives, as well as their security properties. One key feature of Tamarin is that it offers an interactive mode when the tool fails to prove a protocol, while ProVerif typically offers more automation.

ProVerif has been developed for 20 years and has been used to analyze hundreds of protocols, including major deployed protocols such as TLS [BBK17], Signal [KBB17], Noise [Per16, KNB19], the avionic protocol Arinc823 [Bla17], and the Neuchâtel voting protocol [CGT18]. ProVerif is taught in several universities (in specialized Masters) and summer schools. The tool can analyze a large class of security properties, either specified as correspondence properties (for example requesting that an event occurs before another one) or as

equivalence properties, which state that an attacker should not be able to distinguish between two scenarios. Correspondence properties can be used to specify authentication, consistent views between parties, or verifiability properties, while equivalence properties are often used to specify privacy properties like anonymity, non traceability, or vote privacy. Given a protocol and a security property, ProVerif may either prove that the property is satisfied or exhibit an attack. It may also return "cannot be proved" meaning that it can not reach a conclusion. Finally, it may be that ProVerif is not efficient enough to conclude in a reasonable amount of time, or that ProVerif does not terminate at all.

Our contributions. We have carried out a major overhaul of ProVerif, improving its precision, its efficiency, its expressiveness, and introducing some level of interactivity by allowing users to declare intermediate properties helping ProVerif to complete proofs. In more details, our contributions can be summarized as follows:

- support for axioms, lemmas, and restrictions as in Tamarin, in order to obtain the best of the two tools: a high level of automation as well as the possibility to interact with the tool. Lemmas specify intermediate properties meant to help the proof. Axioms are similar to lemmas but do not need to be proved since they are typically guaranteed by other means (e.g. proof by hand). Restrictions are a convenient modeling technique to exclude behaviors that cannot occur in practice (e.g. concurrent access to a lock state).

- improved precision: ProVerif returns "cannot be proved" less often. We introduce a `precise` option that automatically generates (sound) axioms that refine the abstractions made by ProVerif when analyzing a protocol. This helps to conclude that a protocol is either secure or has an attack.

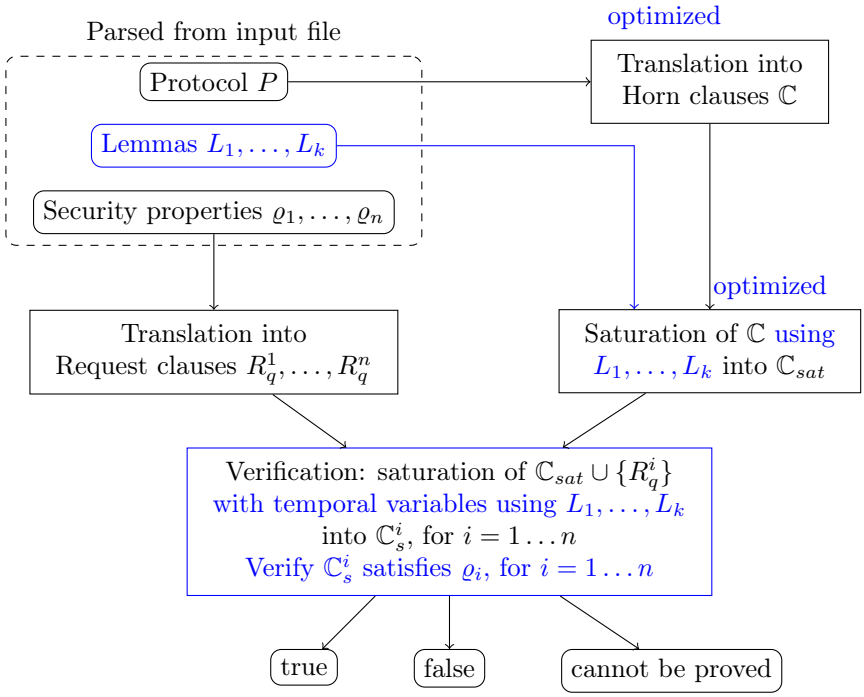
- support for natural numbers together with addition (between integers and at most one variable) and comparison. Natural numbers can be used to specify protocols with counters or can model time evolution.

- support for temporal queries. A query is a security property that ProVerif should prove. Temporal queries are queries in which some events are proved to happen before others, such as $\mathsf{event}(Counter(c_1))@t_1 \wedge \mathsf{event}(Counter(c_2))@t_2 \Rightarrow t_1 > t_2 \vee c_1 \leq c_2$ where we specify that the counter can only increase; the fact $\mathsf{event}(Counter(c))@t$ means that the counter has value c at time t.

- better treatment of injective queries, that is, queries where the occurence of some event (for example, the delivery of some product) can be associated injectively to another event (for example a payment). The injective property ensures here that there are at least as many payments as the number of deliveries. Such queries previously often yield a "cannot be proved".

- major speed improvements. The new ProVerif typically runs 30-40 times faster than ProVerif 2.00 on large protocols and up to exponentially faster on some examples.

As a result, ProVerif can now support more protocols, in particular protocols with global states, for both reachability and equivalence. Global states include counters, cells, tables and are typically difficult to handle for ProVerif due to its internal abstractions. A tool GSVerif was introduced [CCT18] (in the context of reachability properties) to automatically generate



Parts in blue indicate the novelties introduced in this paper.

Figure 1: Overview of the ProVerif procedure.

properties that are proved to hold but cannot be proved by ProVerif. The properties generated by GSVerif can now be stated as axioms, and used at an earlier stage of the procedure, yielding more successful proofs. Moreover, our approach now also applies to equivalence properties. Hence ProVerif can automatically prove equivalence properties for protocols with global states, such as vote privacy in voting protocols that require to maintain a table of all received votes.

Moreover, our experiments show that ProVerif can now prove or disprove (that is, exhibit an attack) in many more protocols of the literature. In terms of efficiency, the analysis of 42 protocols from the Noise Protocol Framework took more than 170h and now takes 20min, hence is at least 516 times faster. The analysis of the Neuchâtel voting protocol is parameterized by the number k of voting options. For $k \geq 3$, the verification of ballot privacy took more than 24h and now takes 3s for $k = 3$ and 4h36min for $k = 6$. Our changes have been integrated in the official distribution of ProVerif (release 2.02pl1 available at `https://proverif.inria.fr`).

A new procedure. All these enhancements and improvements of ProVerif have been obtained through a major rewrite of the internal procedure of the tool. Let us overview the procedure of ProVerif, as summarized in Figure 1. ProVerif first translates protocols into a set $\mathbb{C}$ of Horn clauses, a subclass of first order logic. Then, it saturates the clauses by resolution, yielding a simpler set of clauses $\mathbb{C}_{sat}$ that derive the same facts. If this saturation procedure terminates, then ProVerif verifies the security property by saturating again $\mathbb{C}_{sat}$ with the request clause R obtained by translating the security property, and verifying that the obtained clauses satisfy the property. The correctness of ProVerif ensures that, if this verification succeeds, then the initial property holds. Otherwise, either ProVerif can

reconstruct an attack against the initial protocol following the corresponding clause derivation, or ProVerif cannot conclude and returns "cannot be proved".

It is not easy in this context to add lemmas since they cannot be easily interpreted by ProVerif as an "help". Instead, we modified the internal saturation procedure of ProVerif to refine clauses. Intuitively, given an already proved lemma $A \Rightarrow B$, a clause $H \rightarrow C$ can be replaced by $H \wedge B \rightarrow C$ as soon as H entails A. This yields a more precise clause, possibly helping termination. This is sound as soon as we only use lemmas that have been already proved. However, we also allow a lemma L_i to be proved by induction on the length of the execution trace. We can prove that the first saturation procedure remains sound for such induction thanks to the invariant that facts in hypotheses of clauses always happen before facts in the conclusion, and strictly before for facts that occur in lemmas. However, in the verification step, this property is lost. Therefore, we have entirely revisited the verification procedure, to keep track that certain facts happen (strictly) before others, so that a lemma proved by induction can be used to refine a clause only when the ordering is compatible (hence guaranteeing soundness). More precisely, facts are now annotated with temporal variables and clauses include ordering constraints on these variables. We provide a sound procedure to resolve such clauses. Thanks to these temporal variables, it is then easy to support temporal queries.

The introduction of natural numbers with addition and comparison follows (and generalizes) the approach initiated in [CCT18]: our saturation procedure relies on the Pratt algorithm [Pra77] to solve inequality constraints that appear in clauses.

We prove the correctness of the new procedure, for the entire syntax and semantics of ProVerif. We cover optimizations and features that were never formally defined in previous papers. For instance, we generalize the definition of correspondence queries, which were previously defined only with events in their conclusion.

Contents

2 Model

In this section, we will present the process calculus and the security properties we are focusing on. Note that since this aims to improve the internal algorithm of ProVerif, we will reuse many of the notations and definitions of the papers describing the current algorithm of ProVerif [BAF08, Bla09, Bla16].

2.1 Syntax

We assume a set of variables $\mathcal{V}$, a set of names $\mathcal{N}$. We consider a finite signature Σ of function symbols with their arity partitioned in four sets $\mathcal{F}_d$, $\mathcal{F}_c$, $\mathcal{F}_e$ and $\mathcal{F}_{tbl}$ representing respectively the *constructor*, *destructor* function symbols, *events* and *tables*. The syntax for *terms*, *expressions*, *events*, *predicates* and *processes* is displayed in Figure 2.

$M, N ::=$	terms
x	variable ($x \in \mathcal{V}$)
n	name ($n \in \mathcal{N}$)
$f(M_1, \ldots, M_k)$	applied $f \in \mathcal{F}_c$
$ev ::=$	events
$e(M_1, \ldots, M_k)$	($e \in \mathcal{F}_e$)
$D ::=$	expressions
M	term
$h(D_1, \ldots, D_k)$	applied $h \in \mathcal{F}_d \cup \mathcal{F}_c$
fail	failure
$P, Q ::=$	processes
0	nil
$\mathsf{out}(N, M); P$	output
$\mathsf{in}(N, x); P$	input
$P \mid Q$	parallel composition
$!P$	replication
$\mathsf{new}\ a; P$	restriction
$\mathsf{phase}\ \kappa; P$	phase
$\mathsf{insert}\ tbl(M_1, \ldots, M_n); P$	table insertion
$\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q$	table lookup
$\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q$	assignment
$\mathsf{event}(ev); P$	event

Figure 2: Syntax of the core language of ProVerif.

Terms represents data and can be built as a variable, a name or the application of constructor function symbol on terms. As usual, we define substitutions as functions from variables to terms, and the application of a substitution σ to an expression D is denoted $D\sigma$. The most general unifier of two terms M and N is denoted $mgu(M, N)$.

Destructor function symbols can manipulate terms and only appear in expressions. Their

algebraic properties are expressed by means of rewrite rules. More specifically, the behavior of a destructor function symbol g is modeled by an ordered list of rewrite rules of the form $g(U_1, \ldots, U_n) \to U$ where $U_1, \ldots, U_n, U$ are *may-fail terms*, that are either terms M or the constant fail or a *may-fail variable* u. Note that we may-fail variables are never used in the protocols, only in defining rewrite rules. Typically, a may-fail variable can only be instantiated by fail or a term M.

In Σ, we also associate to each destructor function symbol a list of rewrite rules $\mathsf{def}(g) = [g(U_{i,1}, \ldots, U_{i,n}) \to U_i]_{i=1}^{k}$. The evaluation of an expression is as follows: $g(V_1, \ldots, V_n)$ *evaluates* to V, denoted $g(V_1, \ldots, V_n) \Downarrow V$, when:

- either there exists a substitution σ and $1 \leq i \leq k$ such that $U_i\sigma = V$, for all $j \in \{1, \ldots, n\}$, $V_j = U_{i,j}\sigma$ and for all $i' < i$, for all σ', $(U_{i',1}, \ldots, U_{i',n})\sigma' \neq (V_1, \ldots, V_n)$.
- or else $V = \mathsf{fail}$.

The evaluation of a ground expression follows: $\mathsf{fail} \Downarrow \mathsf{fail}$, $M \Downarrow M$ for all terms M and $h(D_1, \ldots, D_n) \Downarrow U$ when $D_1 \Downarrow U_1$, …, $D_1 \Downarrow U_n$ and

- if $h \in \mathcal{F}_d$ then $h(U_1, \ldots, U_n) \Downarrow U$;
- if $h \in \mathcal{F}_c$ and $U_1, \ldots, U_n$ are terms then $U = h(M_1, \ldots, M_n)$;
- otherwise $U = \mathsf{fail}$.

We natively consider some boolean constants, namely *true* and *false*, as well as the destructors *and*, *or* and *not* with their expected semantics. We also consider a subset of $\mathcal{F}_c$, called *data constructor function symbols* and denoted $\mathcal{F}_{data}$, that are functions symbols that the attacker can always deconstruct. For examples, tuples of any arity are in $\mathcal{F}_{data}$. For all $f/n \in \mathcal{F}_{data}$, we assume the existence of a destructor π_i^f, for $i = 1 \ldots n$, that is the i-th projection of f. Formally, $\mathsf{def}(\pi_i^f) = [\pi_i^f(f(x_1, \ldots, x_n)) \to x_i]$.

Example 1. The standard asymmetric encryption primitives can be modeled by considering a constructor *aenc* of arity 3, a constructor *pk* of arity 1, and a destructor *adec* of arity 2 with the following rewrite rule: $\mathsf{def}(adec) = [adec(aenc(x, y, pk(z)), z) \to x]$. The evaluation of the ground term $adec(aenc(a, r, pk(k)), k)$ applies the rewrite rule, yielding $adec(aenc(a, r, pk(k)), k) \Downarrow a$.

Similarly, signatures can be modeled with the rule $\mathsf{def}(checksign) = [checksign(sign(x, y), vk(y)) \to x]$. This rule combines the verification of the signature and the retrieval of the message whose signature has been checked. ▶

Example 2. Defining the behaviour of a destructor with a sequence of rewrite rules rather than a set of rewrite rules, as it is usually the case in the litterature, intuitively allows to express conditional tests on the application of rewrite rules. One can see the evaluation of a destructor as "Apply the first rewrite rule if applicable, else apply the second rewrite rule if applicable, else …". This is strictly more powerful than standard set of rewrite rules. Consider the destructor *ifelse* of arity 3 such that $ifelse(x, y, z)$ should be rewritten in y when x is *true*, and should be rewritten in z otherwise. This can be expressed in our formalism with the sequence of rewrite rules $\mathsf{def}(ifelse) = [ifelse(true, y, z) \to y; ifelse(x, y, z) \to z]$. Using a set of rewrite rule, the term $ifelse(true, a, b)$ could be both rewritten in a and b (hence yielding a non-convergent rewrite system). Using sequence of rewrite rules, we have only have $ifelse(true, a, b) \Downarrow a$. ▶

Example 3. May-fail variables allow to reason on the success or evaluation of an expression. Consider the sequences $\mathsf{def}(dec)$ and $\mathsf{def}(ifelse)$ defined in Examples 1 and 2 respectively. The destructor $ifelse$ was defined as $ifelse(x, y, z)$ *should be rewritten in* y *when* x *is true, and should rewritten in* z *otherwise.* Hence, the intuition would be that the value of z should not matter in the evaluation of $ifelse(true, y, z)$. This is however not the case when z cannot be evaluated, i.e. when its evaluation fails as shown below:

$$ifelse(true, a, b) \Downarrow a \qquad \text{but} \qquad ifelse(true, a, dec(b, c)) \Downarrow \mathsf{fail}$$

These evaluation failures can be captured by using may-fail variables in the rewrite rules: by defining $\mathsf{def}(ifelse) = [ifelse(true, u, v) \to u; ifelse(x, u, v) \to v]$ where x is a variable and u, v are may-fail variables, we obtain $ifelse(true, a, dec(b, c)) \Downarrow a$. ▶

Event function symbols $e \in \mathcal{F}_e$ can be applied on terms to build events $e(M_1, \ldots, M_n)$. They are used to express correspondence security properties and can be executed by processes with the construct $\mathsf{event}(e(M_1, \ldots, M_n)); P$. The other constructs in processes are standard. The process $\mathsf{in}(N, x); P$ models the input on the channel N of a term that is bound to x when executing P. The process $\mathsf{out}(N, M); P$ represents the output of the term M on the channel N. The process $P \mid Q$ models the concurrent execution of P and Q. The replication $!P$ represents an unbounded number of copies of P. The restriction $\mathsf{new}\ a; P$ generates a fresh name a that can be used in P. Finally, the construct $\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q$ evaluates the expression D, executing P with x bound to M when D evaluates to a message M, and otherwise executing Q.

Natural numbers Following the work of [CCT18], we also consider natural numbers using the Peano representation. We assume that $zero/0$ and $succ/1$ are two function symbols in $\mathcal{F}_c$. Note that for simplicity, we will denote by $x + n$ the term $succ^n(x)$, and we will denote by $n \in \mathbb{N}$ the term $succ^n(zero)$. We say that a ground term M is a natural number if $M = n$ for some $n \in \mathbb{N}$. The symbol $succ$ is a data constructor function symbol, meaning that we consider a destructor $minus/1$ defined as $\mathsf{def}(minus) = [minus(succ(x)) \to x]$.

Moreover, we consider two special additional functions $nat/1$ and $geq/2$ whose evaluation on ground term is defined as follows:

- $nat(D) \Downarrow true$ (resp. $false$) if $D \Downarrow M \in \mathbb{N}$ (resp. $M \notin \mathbb{N}$). Otherwise $nat(D) \Downarrow \mathsf{fail}$.
- $geq(D_1, D_2) \Downarrow true$ (resp. $false$) if for $i = 1, 2$, $D_i \Downarrow M_i \in \mathbb{N}$ and $M_1 \geq M_2$ (resp. $M_1 < M_2$). Otherwise $geq(D_1, D_2) \Downarrow \mathsf{fail}$.

2.2 Semantics

A *configuration* $\kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ is given by a multiset $\mathcal{P}$ of processes, representing the current state of the process, a set of names $\mathcal{E}$ representing the free names of $\mathcal{P}$ and the names created by the adversary, the current phase κ, a set $\mathcal{T}$ of terms of the form $tbl(M_1, \ldots, M_n)$ representing the elements inserted into tables, and a set $\mathcal{A}$ of terms known by the adversary. The semantics of processes is defined through a reduction relation $\xrightarrow{\ell}_o$ between configuration, defined in fig. 3, where ℓ is either the empty label or a label of the form $\mathrm{msg}(M, N)$ or $\mathrm{event}(ev)$ with M, M being terms and ev being an event.

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{0\}\!\}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \quad \text{(Nil)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P \mid Q\}\!\}, \mathcal{T}, \mathcal{A} \to_o \mathcal{E}, \mathcal{P} \cup \{\!\{P, Q\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{(Par)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{!P\}\!\}, \mathcal{T}, \mathcal{A} \to_o \mathcal{E}, \mathcal{P} \cup \{\!\{P, !P\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{(Repl)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{new}\ a; P\}\!\}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E} \cup \{a'\}, \mathcal{P} \cup \{\!\{P\{^{a'}/_{a}\}\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{if } a' \notin \mathcal{E} \quad \text{(Restr)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{out}(N, M); P, \mathsf{in}(N, x); Q\}\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P, Q\{^{M}/_{x}\}\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{(I/O)}$$

$$\kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_o \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \quad \text{if } N, M \in \mathcal{A} \quad \text{(Msg)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P\{^{M}/_{x}\}\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{if } D \Downarrow M \quad \text{(Let1)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{Q\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{if } D \Downarrow \mathsf{fail} \quad \text{(Let2)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{out}(N, M); P\}\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}, \mathcal{T}, \mathcal{A} \cup \{M\} \quad \text{if } N \in \mathcal{A} \quad \text{(Out)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{in}(N, x); Q\}\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{Q\{^{M}/_{x}\}\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{if } N, M \in \mathcal{A} \quad \text{(In)}$$

$$\kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \cup \{M\} \quad \text{(App)}$$
$$\text{if } M_1, \ldots, M_n \in \mathcal{A},\ f/n \in \mathcal{F}_c \cup \mathcal{F}_d \text{ and } f(M_1, \ldots, M_n) \Downarrow M$$

$$\kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \to_o \kappa, \mathcal{E} \cup \{a'\}, \mathcal{P}, \mathcal{T}, \mathcal{A} \cup \{a'\} \quad \text{if } a' \notin \mathcal{E} \quad \text{(New)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{event}(ev); P\}\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{event}(ev)}_o \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}, \mathcal{T}, \mathcal{A} \quad \text{(Event)}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \mathcal{P}_{>\kappa+1} \cup \{\!\{\mathsf{phase}\ \kappa + 1; P_i\}\!\}_{i=1}^{k}, \mathcal{T}, \mathcal{A} \to_o \kappa + 1, \mathcal{E}, \mathcal{P}_{>\kappa+1} \cup \{\!\{P_i\}\!\}_{i=1}^{k}, \mathcal{T}, \mathcal{A} \quad \text{(Phase)}$$
$$\text{if all processes of } \mathcal{P} \text{ do not start with some } \mathsf{phase}\ \kappa',\ \kappa' > \kappa$$
$$\text{and all processes of } \mathcal{P}_{>\kappa+1} \text{ start with some } \mathsf{phase}\ \kappa',\ \kappa' > \kappa + 1$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{insert}\ tbl(M_1, \ldots, M_n); P\}\!\}, \mathcal{T}, \mathcal{A} \to_o \quad \text{(Insert)}$$
$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P\}\!\}, \mathcal{T} \cup \{tbl(M_1, \ldots, M_n)\}, \mathcal{A}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \mathcal{T}, \mathcal{A} \to_o \quad \text{(Get1)}$$
$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{P\sigma\}\!\}, \mathcal{T}, \mathcal{A}$$
$$\text{if } tbl(x_1, \ldots, x_n)\sigma \in \mathcal{T} \text{ and } D\sigma \Downarrow \mathit{true}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \mathcal{T}, \mathcal{A} \to_o \quad \text{(Get2)}$$
$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\{Q\}\!\}, \mathcal{T}, \mathcal{A}$$
$$\text{if for all } \sigma,\ tbl(x_1, \ldots, x_n)\sigma \notin \mathcal{T} \text{ or } \neg(D\sigma \Downarrow \mathit{true})$$

Figure 3: Transitions between configurations.

An initial configuration is a closed configuration of the form $0, \mathcal{E}, \{\!\{P\}\!\}, \emptyset, \mathcal{A}$ where $\mathcal{A}$ contains only names and $\mathcal{E}$ is the union of $\mathcal{A}$ with the free names of P. We will denote such initial configuration $\mathcal{E}, P, \mathcal{A}$.

The reduction rule Nil removes the process 0 from the multiset since it does nothing, the rule Par apply the parallel composition and the rule Repl duplicates the process P hence modeling replication. The rule Restr generates new private names hence the condition $a' \notin \mathcal{E}$. The rule I/O allows communication between an output and input. The rule Msg indicates that the attacker is sending the message M on the channel N without interacting with the honest process. The rules Let1 and Let2 define the semantics of the evaluation of an expression D. Note that the condition $D \Downarrow M$ expresses the fact that the evaluation of D succeeded since M is a term hence not the expression constant fail. The rule Event executes the event M.

Remark 1. The rule Msg in itself is not necessary for the execution of the protocol. However, it allows us to have a a strict correspondence with the original semantics of applied pi calculus, that is an *operational semantics*. Such a semantics operates only on process and not on configuration. In particular, the attacker actions are also described by a process. A security property of some term M on a process P thus corresponds to checking that for all attacker process P_a, $P_a \mid P$ does not reveal M. In the next section, we will see that we allow correspondence query to check wether a message M was sent on a channel N. In the operational semantics, this would be possible when the communication occurs on within P (corresponding to the rule I/O) or between P_a and P (corresponding to the rules Out and In) or within P_a (corresponding to the rule Msg). ▶

A trace of a configuration $\mathcal{C}_0$ is a finite sequence $\mathcal{C}_0 \xrightarrow{\ell_1}_o \mathcal{C}_1 \xrightarrow{\cdots}_o \xrightarrow{\ell_n}_o \mathcal{C}_n$.

Note that the syntax of processes does not explicitly include a conditional of the form $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$. This is because it can be modeled by an assignment $\mathsf{let}\ x = \mathit{equals}(M, N)\ \mathsf{in}\ P\ \mathsf{else}\ Q$ where x is fresh and $\mathit{equals}/2$ is a destructor function symbol defined by $\mathsf{def}(\mathit{equals}) = [\mathit{equals}(x, x) \to x]$. We therefore will assume from now on that $\mathcal{F}_d$ contains $\mathit{equals}/2$.

As we will see in the rest of this section, we aim to improve the verification of both correspondence and equivalence queries. However, in order to explain how the algorithms work, we need to define another semantics, the associated correspondence and equivalence queries and finally show the soundness of this semantics w.r.t. to this main semantics. Therefore, to avoid redefining every notion multiple times, we will introduce general definitions that we will reuse in both semantics.

Generally, given S an infinite set, given $\to$ a labeled binary relation on S and given $A_0 \in S$, we denote by $\mathsf{trace}(A_0, \to) = \{A_0 \xrightarrow{\ell_1} A_1 \xrightarrow{\ell_2} \dots \xrightarrow{\ell_n} A_n\}$ the set of traces from A_0 by $\to$. Moreover, given a trace $T \in \mathsf{trace}(A_0, \to)$, if $T = A_0 \xrightarrow{\ell_1} A_1 \to \dots \xrightarrow{\ell_n} A_n$, we say that T contains n *steps*. Given $\tau \in \{0, \dots, n\}$, we denote by $T[\tau]$ the configuration A_0. Furthermore, we call the *τ-th step of T* the transition $T[\tau - 1] \xrightarrow{\ell_\tau} T[\tau]$. Finally, we define $\mathsf{max}_{\mathit{step}}(T) = n$, $\mathit{steps}(T) = \{0, \dots, n\}$ and $\mathbb{T}(\to) = \bigcup_{A \in S} \mathsf{trace}(A, \to)$ the set of all traces by $\to$.

Example 4. Belenios [CGG19] is a simple voting protocol used in more than 200 elections each year. Each voter sends her vote encrypted with the public key of the election, and signed with a credential. The encrypted vote is then published on a bulletin board. At the end of the election, ballots are shuffled (using mixnets) and votes are published in a random order. Other variants of Belenios exist with homomorphic encryption but we present here only a simplified version, omitting for example the zero-knowledge proofs of correct decryption.

The behavior of a voter can be represented by the following process:

$V(\mathit{vote}, \mathit{sk}) = \mathsf{in}(c, x_r);\mathsf{new}\ r;$
$\quad \mathsf{out}(c, \mathit{sign}(\mathit{aenc}(\mathit{vote}, (r, x_r), \mathit{pk}(\mathit{sk}_e)), \mathit{sk}))$

This models the fact that the voting client uses both its own randomness and some external randomness (typically from the server) to encrypt the vote.

For simplicity, we model the voting server together with the tally phase. The server receives votes, checks the validity of signatures and once the election is over, publishes the decrypted ballots in a random order.

Board=

$\mathsf{in}(c, x_1); \mathsf{let}\ y_1 = \mathit{checksign}(x_1, \mathit{vk}(sk_a))\ \mathsf{in}$
$\mathsf{in}(c, x_2); \mathsf{let}\ y_2 = \mathit{checksign}(x_2, \mathit{vk}(sk_b))\ \mathsf{in}$
$\mathsf{in}(c, x_3); \mathsf{let}\ y_3 = \mathit{checksign}(x_3, \mathit{vk}(sk_c))\ \mathsf{in}$
$\quad \mathsf{out}(c, \mathit{adec}(y_1, sk_e))$
$\quad \mid \mathsf{out}(c, \mathit{adec}(y_2, sk_e))$
$\quad \mid \mathsf{out}(c, \mathit{adec}(y_3, sk_e))$

The fact that the decrypted ballots are sent in a random order is modeled here by considering three outputs in parallel: the adversary cannot distinguish the order of the output.

Then the process for modeling Belenios altogether is:

$$P_{Bel} = V(v_a, sk_a) \mid V(v_b, sk_b) \mid \mathit{Board} \mid \mathit{Setup}$$

where *Setup* is defined as:

$$\mathsf{out}(c, \mathit{vk}(sk_a)) \mid \mathsf{out}(c, \mathit{vk}(sk_b)) \mid \mathsf{out}(c, \mathit{pk}(sk_e))$$

P_{Bel} models a system with two honest voters and a dishonest one, whose secret key is sk_c. The initial configuration is $\mathcal{C}_0 = \mathcal{E}_0, P_{Bel}, \mathcal{A}_0$ with $\mathcal{E}_0 = \{c, sk_a, sk_b, sk_c, sk_e, v_a, v_b\}$ and $\mathcal{A}_0 = \{c, sk_c, v_a, v_b\}$ modeling the fact that the key sk_c is known to the attacker while the other ones are initially secret. The vote values v_a, v_b are also given to the attacker.

Then a possible execution trace is

$$\mathcal{C}_0 \xrightarrow{\mathrm{msg}(c,\mathit{vk}(sk_a))}_o \xrightarrow{\mathrm{msg}(c,\mathit{vk}(sk_b))}_o \xrightarrow{\mathrm{msg}(c,\mathit{pk}(sk_e))}_o \mathcal{C}_1$$
$$\xrightarrow{\mathrm{msg}(c,r'_a)}_o \xrightarrow{\mathrm{msg}(c,u_a)}_o \xrightarrow{\mathrm{msg}(c,r'_b)}_o \xrightarrow{\mathrm{msg}(c,u_b)}_o \mathcal{C}_2$$

where we omit steps with no labels and $u_\alpha = \mathit{sign}(\mathit{aenc}(v_\alpha, (r_\alpha, r'_\alpha), \mathit{pk}(sk_e)), sk_\alpha)$, $\alpha \in \{a, b\}$ and $\mathcal{C}_2 = \mathcal{E}_2, P_2, \mathcal{A}_2$ with $\mathcal{E}_2 = \mathcal{E}_0 \cup \{r_a, r'_a, r_b, r'_b\}$, $\mathcal{A}_2 = \mathcal{A}_0 \cup \{\mathit{vk}(sk_a), \mathit{vk}(sk_b), \mathit{pk}(sk_e), r'_a, u_a, r'_b, u_b\}$, and $P_2 = \mathit{Board}$. This trace corresponds to the emission of the initial messages of the *Setup* process, followed by the execution of the processes of the two honest voters.

Then more interestingly, instead of casting a standard ballot, the attacker may copy Alice's ballot and sends it on behalf of the dishonest voter. This behavior is reflected by the trace $\mathcal{C}_2 \xrightarrow{\mathrm{msg}(c,u_a)}_o \xrightarrow{\mathrm{msg}(c,u_b)}_o \xrightarrow{\mathrm{msg}(c,u_c)}_o \mathcal{C}_3$. The board first receives the two honest ballots, and then the adversarial one: $u_c = \mathit{sign}(\mathit{checksign}(u_a, \mathit{vk}(sk_a)), sk_c) = \mathit{sign}(\mathit{aenc}(v_1, (r_a, r'_a), \mathit{pk}(k))), sk_c)$.

As noticed in [CS13] in the context of the Helios protocol, this attack yields to a privacy attack. Indeed, the outcome of the election will be $\{v_a, v_a, v_b\}$, hence the attacker can deduce that Alice voted v_a.

To avoid this attack, the bulletin board should never accept two identical encrypted votes. This can be modeled in the process *Board* by checking that y_1, y_2, and y_3 are pairwise distinct, yielding a process Board'. ▶

2.3 Security properties

In this paper, we focus on the three main way to express security properties, namely secrecy, correspondence and equivalence properties.

2.3.1 Correspondence and secrecy properties

Atomic formulas. To express secrecy and correspondence properties, we consider *facts*, *temporal facts* and *atomic formulas* whose syntax is given by the following grammar:

$pev ::=$	event
$\quad \text{event}(e(M_1, \ldots, M_n))$	$\quad$ the event $e(M_1, \ldots, M_n)$ is executed, $e \in \mathcal{F}_e$
$\quad \text{inj}_k\text{-event}(e(M_1, \ldots, M_n))$	$\quad$ the injective event $e(M_1, \ldots, M_n)$ is executed, $e \in \mathcal{F}_{ie}, k \in \mathbb{N}$
$F ::=$	fact
$\quad pev$	$\quad$ an event
$\quad \text{att}_\kappa(M)$	$\quad$ the attacker knows M at phase κ
$\quad \text{msg}_\kappa(M, N)$	$\quad$ the message N was sent on the channel M at phase κ
$\quad \text{table}_\kappa(tbl(M_1, \ldots, M_n))$	$\quad$ the messages $M_1, \ldots, M_n$ are registered in the table tbl at phase κ
$\mathsf{F} ::=$	temporal fact
$\quad F$	$\quad$ a fact
$\quad F@i$	$\quad$ a fact F at step i
$\phi ::=$	generic formula
$\quad M = N$	$\quad$ an equality
$\quad M \neq N$	$\quad$ a disequality
$\quad M \geq N$	$\quad$ an inequality
$\quad \text{isnat}(M)$	$\quad$ M is a natural number
$\quad \neg\text{isnat}(M)$	$\quad$ M is not a natural number
$\alpha ::=$	atomic formula
$\quad \mathsf{F}$	$\quad$ a temporal fact
$\quad \phi$	$\quad$ a generic predicate
$\psi, \psi' ::=$	query conclusion
$\quad \top$	$\quad$ true
$\quad \bot$	$\quad$ false
$\quad \psi \wedge \psi'$	$\quad$ a conjunction
$\quad \psi \vee \psi'$	$\quad$ a disjunction
$\quad \alpha$	$\quad$ an atomic formula
$\quad pev \rightsquigarrow \psi$	$\quad$ a nested query
$\quad pev@i \rightsquigarrow \psi$	$\quad$ a temporal nested query
$\varrho ::=$	correspondence query
$\quad \mathsf{F}_1 \wedge \ldots \wedge \mathsf{F}_n \Rightarrow \psi$	

In injective events $\text{inj}_k\text{-event}(ev)$, the index k represents an occurrence. More specifically, two injective events $\text{inj}_k\text{-event}(ev_1)$ and $\text{inj}_k\text{-event}(ev_2)$ with same index k in a query enforce that the instances of ev_1 and ev_2 should occur at the same step.

Note that for attacker, message and table facts, we may simply write $\text{att}(M)$, $\text{msg}(M, N)$ and $\text{table}(tbl(M_1, \ldots, M_n))$ when there is no phase in the protocol. Given a fact F, we denote by $pred(F)$ the predicate of this fact (e.g. $pred(\text{att}_i(M)) = \text{att}_i$).

Temporal facts One of our contributions is the introduction of temporal facts in correspondence query using the construct $F@i$ where i is a variable, called *temporal variable*. It indicates that the fact F is satisfied by the trace at the step i, *e.g.* $\text{event}(Counter(c_1))@i$. Traditional correspondence queries as defined in ProVerif 2.00 only allow to order the facts of the premise of the query w.r.t. the facts in its conclusion. The *nested queries* included in ProVerif 2.00 permits a limited comparison between facts in the conclusion of the query. Temporal correspondence queries can be seen as a generalization of correspondence queries allowing to order facts occurring anywhere in the query by comparing temporal variables through inequalities, disequalities and equalities in the conclusion of the query. We impose some restrictions on the temporal variables occurring in a query.

Definition 1. *A* temporal correspondence query *is a correspondence query such that:*

- *temporal variables can only occur in generic formulas of the form $i = j$, $i \neq j$ and $i \geq j$ where both i and j are temporal variables (*e.g. *$i+3 \geq j$, $i > 5$ and* $\text{event}(Counter(i))@i$ *are not valid atomic formulae when i,j are temporal variables).*
- *temporal variables can only be bound by at most one fact.*

An atemporal correspondence query *is a correspondence query that does not contain temporal facts $F@i$.*

In the rest of the paper, when we want to talk about a fact F occurring in a query that is not necessarily a temporal fact, we will write $F[@i]$, i.e. $@i$ is optional.

Satisfaction relation We will consider several transition relations $\mathbb{T}(\rightarrow)$ in the rest of the paper. We have already defined $\rightarrow_o$ between configurations. We will consider a variant $\rightarrow_i$, that keeps track more carefully of replications. To prove equivalence properties, we will define similar relations $\rightarrow_{o'}$ and $\rightarrow_{i'}$ between pairs of processes. To avoid duplicating the definitions, some notions will be defined w.r.t. any binary relation $\rightarrow$.

Given a binary labeled relation $\rightarrow$ on some infinite set S, a satisfaction relation on ground atomic formula for $\rightarrow$ is a ternary relation $\vdash$, denoted $T, \tau \vdash \alpha$, where $T \in \mathbb{T}(\rightarrow)$, τ is a step of T and α is a ground atomic formula such that if α is a generic predicate then the satisfiability is independent of T and τ. In such a case, we will just denote $\vdash \alpha$.

For configurations and the transition relation $\rightarrow_o$, we define the following satisfaction relation $\vdash_o$.

Definition 2. *Let $T \in \mathbb{T}(\rightarrow_o)$. Let $\tau \in steps(T)$. We define the satisfaction relation $\vdash_o$ on ground atomic formulas as follows:*

- $T, \tau \vdash_o \text{att}_\kappa(M)$ *if and only if* $T[\tau] \rightarrow_o^* \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ *by only the rules* App, New *and* Phase *such that* $M \in \mathcal{A}$.
- $T, \tau \vdash_o \text{event}(ev)$ *(resp.* $\text{inj}_k\text{-event}(ev)$*) if and only if* $T[\tau - 1] \xrightarrow{\text{event}(ev)}_o T[\tau]$.
- $T, \tau \vdash_o \text{msg}_\kappa(N, M)$ *if and only if* $T[\tau - 1] \xrightarrow{\text{msg}(N,M)}_o T[\tau]$ *with* $T[\tau] = \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$.

- $T, \tau \vdash_o \text{table}_\kappa(tbl(M_1, \dots M_n))$ *if and only if* $T[\tau] = \kappa', \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ *with* $\kappa' \leq \kappa$ *and* $tbl(M_1, \dots, M_n) \in \mathcal{T}$.
- $T, \tau \vdash_o F@\tau$ *if and only if* $T, \tau \vdash_o F$.
- $T, \tau \vdash_o M = N$ *(resp.* $M \neq N$*) if and only if* $M = N$ *(resp.* $M \neq N$*).*
- $T, \tau \vdash_o M \leq N$ *if and only if* $M, N \in \mathbb{N}$ *and* $M \leq N$ *(in their natural number representations).*
- $T, \tau \vdash_o \text{isnat}(M)$ *if and only if* $M \in \mathbb{N}$.
- $T, \tau \vdash_o \neg\text{isnat}(M)$ *if and only if* $M \notin \mathbb{N}$.

Note that $T, \tau \vdash_o \text{att}_\kappa(M)$ holds even if M is not in the attacker's knowledge $\mathcal{A}$ of the last configuration $\mathcal{C}'$ reached by T. Indeed, we wish $T, \tau \vdash_o \text{att}_\kappa(M)$ to hold as soon as the attacker *may deduce* M and even if M is not already explicitly in his knowledge. This is why we consider any evolution $\mathcal{C}' \to_o^* \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ that only uses attacker rules (App), nonce generation (New) and phase progression (Phase).

Example 5. A query $\text{inj}_1\text{-event}(A(x)) \Rightarrow \text{inj}_2\text{-event}(B(x))$ requires that for any trace T, the number of executions of $\mathsf{event}(A(M))$ is smaller than the number of executions of $\mathsf{event}(B(M))$, i.e. to each execution $\mathsf{event}(A(M))$ in T, we can associate a different execution $\mathsf{event}(B(M))$ in T. ▶

A nested query $pev \rightsquigarrow \psi$ typically indicates that both pev and ψ hold and all facts of ψ (e.g. events) are true *before (w.r.t. the trace)* the event pev. Note that it is different from the implication symbol $\Rightarrow$ that really corresponds to a logical implication.

Example 6. Consider the queries $\varrho_1 = (\text{event}(C) \Rightarrow \text{event}(B) \rightsquigarrow \text{event}(A))$ and $\varrho_2 = (\text{event}(C) \wedge \text{event}(B) \rightsquigarrow \text{event}(A))$. In words, the query ϱ_1 intuitively indicates that *if the event C was executed then both events B and A must also be executed and A must have been executed before B*. On the other hand, the query ϱ_2 intuitively indicates that *if the events C and B are executed then the event A must also be executed.*

Consider three traces T_1, T_2, T_3 with $T_1[0] \xrightarrow{\text{event}(A)}_o T_1[1] \xrightarrow{\text{event}(B)}_o T_1[2] \xrightarrow{\text{event}(C)}_o T_1[3]$, $T_2[0] \xrightarrow{\text{event}(A)}_o T_2[1] \xrightarrow{\text{event}(C)}_o T_2[2]$ and $T_3[0] \xrightarrow{\text{event}(C)}_o T_3[1] \xrightarrow{\text{event}(A)}_o T_3[2] \xrightarrow{\text{event}(B)}_o T_3[3]$. On these specific traces, ϱ_1 holds on T_1 and T_3 but not on T_2. On the other hand, ϱ_2 holds on all three traces.

Notice that on T_3, even though the event C is executed before the events A and B, both queries hold on T_3. This is due to the fact that $\Rightarrow$ has no temporal requirement and only acts as a logical implication, in contrast to $\rightsquigarrow$. Of course, when proving a query on a process, we check that the query holds on *all* traces, meaning that the query must hold on T_3 but also on any prefix of T_3, which is not the case here. ▶

Example 7. The temporal query $\text{event}(Counter(c_1))@i \wedge \text{event}(Counter(c_2))@j \Rightarrow i > j \vee c_1 \leq c_2$ intuitively indicates that the values emitted in the event *Counter* can only increase during the execution. ▶

Let us denote $\mathcal{S}_\sigma$ the set of all ground substitutions. In order to formally define the notion of injectivity in a query $\bigwedge_{i=1}^n \mathsf{F}_i \Rightarrow \psi$, for all traces of the protocol, we will associate to each fact and in particular each $\text{inj}_k\text{-event}(ev)$ in ψ a partial function μ from $steps(T)^n \times \mathcal{S}_\sigma$ to

steps(T). Intuitively, this partial function specifies how we associate executions of $F_1, \ldots, F_n$ to an execution of inj$_k$-event(M). For example, if $F_1, \ldots, F_n$ have been executed in T at step $\tau_1, \ldots, \tau_n$ respectively and have been instantiated by the substitution σ then the event event(ev) (resp. inj$_k$-event(ev)) should be executed in T at step $\mu((\tau_1, \ldots, \tau_n), \sigma)$. Note that we will associate to each event and injective event in ψ a different function μ.

Formally, given an integer n, an *annotated query conclusion w.r.t. n* is a formula Ψ defined as follows:

$$\begin{array}{rl} \mu, \mu', \ldots ::= & \text{partial functions from } \mathit{steps}(T)^n \times \mathcal{S}_\sigma \text{ to } \mathit{steps}(T) \text{ such that} \\ & \text{for all } \tilde{\tau} \in \mathit{steps}(T)^n, \text{ for all } \sigma \in \mathcal{S}_\sigma, \text{ if } \mu(\tilde{\tau}, \sigma) \text{ is defined} \\ & \text{then } \mu(\tilde{\tau}, \sigma) \leq \max(\tilde{\tau}) \\ \\ \Psi, \Psi' ::= & \top \mid \bot \mid \Psi \wedge \Psi' \mid \Psi \vee \Psi' \mid \phi \mid \mathsf{F}^\mu \mid \mathsf{F}^\mu \rightsquigarrow \Psi \end{array}$$

Given an annotated query conclusion Ψ, we denote by $\overline{\Psi}$ the formula obtained from Ψ in which all partial functions are removed.

Example 8. Consider the following correspondence query.

$$\text{inj}_1\text{-event}(A(x)) \wedge \text{event}(B(y)) \Rightarrow \text{inj}_2\text{-event}(C(x)) \wedge \text{event}(D(y))$$

Any formula $\Psi = \text{inj}_2\text{-event}(C(x))^{\mu_1} \wedge \text{event}(D(y))^{\mu_2}$ with μ_1, μ_2 two partial functions from $\{0, \ldots, m\}^2 \times \Sigma$ to $\{0, \ldots, m\}$ is an annotated query conclusion w.r.t. $(2, m)$ and $\overline{\Psi} = \text{inj}_2\text{-event}(C(x)) \wedge \mathsf{event}(D(y))$. ▶

Definition 3. *Let $n \in \mathbb{N}$ and let Ψ be a ground annotated query conclusion w.r.t. n. Let $\rightarrow$ be a binary relation.*

A valuation of Ψ is a tuple $(\vdash, T, (\tilde{\tau}, \sigma))$ where $\vdash$ is a satisfaction relation on ground atomic formulas for $\rightarrow$, $T \in \mathbb{T}(\rightarrow)$, $\tilde{\tau} \in \mathit{steps}(T)^n$ and $\sigma \in \mathcal{S}_\sigma$. The satisfaction relation $\models$ for the formula Ψ, denoted $(\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi$, is defined as follows:

$$\begin{array}{lll} (\vdash, T, (\tilde{\tau}, \sigma)) \models \phi & \textit{iff} & \vdash \phi \\ (\vdash, T, (\tilde{\tau}, \sigma)) \models \mathsf{F}^\mu & \textit{iff} & \mu(\tilde{\tau}, \sigma) \textit{ is defined and } T, \mu(\tilde{\tau}, \sigma) \vdash \mathsf{F} \\ (\vdash, T, (\tilde{\tau}, \sigma)) \models \mathsf{F}^\mu \rightsquigarrow \Psi & \textit{iff} & (\vdash, T, (\tilde{\tau}, \sigma)) \models \mathsf{F}^\mu \wedge \Psi \textit{ and} \\ & & \textit{for all } \mathsf{F}'^{\mu'} \textit{ in } \Psi, \ \mu'(\tilde{\tau}, \sigma) \textit{ defined implies } \mu'(\tilde{\tau}, \sigma) \leq \mu(\tilde{\tau}, \sigma) \\ (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi \wedge \Psi' & \textit{iff} & (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi \textit{ and } (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi' \\ (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi \vee \Psi' & \textit{iff} & (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi \textit{ or } (\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi' \\ (\vdash, T, (\tilde{\tau}, \sigma)) \models \top & & \end{array}$$

We can now formally define the satisfiability of a correspondence query.

Definition 4. *Let $\varrho = (\bigwedge_{i=1}^n \mathsf{F}_i \Rightarrow \psi)$ be a temporal correspondence query. Let $\rightarrow$ be a binary relation.*

For all satisfaction relations $\vdash$ on ground atomic formulas for $\rightarrow$, for all $\mathcal{S} \subseteq \mathbb{T}(\rightarrow)$, we say that $\vdash, S$ satisfy ϱ, denoted $(\vdash, \mathcal{S}) \models \varrho$ if and only if for all $T \in \mathcal{S}$, there exists an annotated query conclusion Ψ w.r.t. n such that $\overline{\Psi} = \psi$ and:

1. *for all tuple of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, for all substitutions σ, if $T, \tau_j \vdash \mathsf{F}_j\sigma$ for $j = 1 \ldots n$ then there exists σ' such that $\mathsf{F}_j\sigma = \mathsf{F}_j\sigma'$ for $j = 1 \ldots n$ and $(\vdash, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma'$.*

2. *for all* $\text{inj}_k\text{-event}(ev)[@i]^{\mu}$ *occurring in* Ψ, $\mu(\tilde{\tau}, \sigma) = \mu(\tilde{\tau}', \sigma')$ *implies that* $\tau_j = \tau'_j$ *for all* j *such that* $F_j = \text{inj}_{k_j}\text{-event}(ev_j)$ *for some* k_j, ev_j.

3. *for all* $\text{inj}_{k_1}\text{-event}(ev_1)[@i_1]^{\mu_1}$ *and* $\text{inj}_{k_2}\text{-event}(ev_2)[@i_2]^{\mu_2}$ *occurring in* Ψ, $k_1 = k_2$ *implies* $\mu_1 = \mu_2$

The first bullet point of definition 4 intuitively checks that the events in the query are always executed before $\mathsf{F}_1, \ldots, \mathsf{F}_n$. The second bullet point checks the injective requirements of the query are satisfied. Finally, the third bullet ensures that injective events with the same injective index are associated with the same partial function. Injective indices are very useful as they ensure that correspondence queries are stable by boolean distributivity.

Lemma 1. *Let* $\varrho = (\bigwedge_{i=1}^{n} \mathsf{F}_i \Rightarrow \psi[\psi_A \wedge (\psi_B \vee \psi_C)])$. *For all satisfaction relation* $\vdash$ *on ground atomic formulas for* $\rightarrow$, *for all* $\mathcal{S} \subseteq \mathbb{T}(\rightarrow)$,

$$(\vdash, \mathcal{S}) \models \varrho \text{ if and only if } (\vdash, \mathcal{S}) \models \bigwedge_{i=1}^{n} \mathsf{F}_i \Rightarrow \psi[(\psi_A \wedge \psi_B) \vee (\psi_A \vee \psi_C)]$$

Thanks to lemma 1, we can always transform a query $\mathsf{F}_1 \wedge \ldots \wedge \mathsf{F}_n \Rightarrow \psi$ into a query where ψ is in *disjunctive normal form* (DNF), that is i.e. $\psi = \bigvee_i \bigwedge_j \psi_{i,j}$ where all $\psi_{i,j}$ are either facts, generic predicates or nested queries of the form $\mathsf{F} \rightsquigarrow \psi'$ with ψ' in DNF.

Remark 2. Our definition of correspondence query is in fact a generalization of the one in [Bla09]. First, they only consider queries directly in disjunctive normal form. Second, injective events do not have injective indices. Both generalizations are in fact closely related. Indeed, the satisfaction relation defined in [Bla09] corresponds to our definition when all the injective indices are disjoints. However in such a case, the satisfaction is not stable anymore by boolean distributivity.

For example, the query $\text{inj}_1\text{-event}(A) \Rightarrow \text{inj}_2\text{-event}(B) \wedge (\text{event}(C) \vee \text{event}(D))$ is equivalent to $\text{inj}_1\text{-event}(A) \Rightarrow (\text{inj}_2\text{-event}(B) \wedge \text{event}(C)) \vee (\text{inj}_2\text{-event}(B) \wedge \text{event}(D))$ but is not equivalent to $\text{inj}_1\text{-event}(A) \Rightarrow (\text{inj}_2\text{-event}(B) \wedge \text{event}(C)) \vee (\text{inj}_3\text{-event}(B) \wedge \text{event}(D))$. ▶

Remark 3. In an input file of this paper's ProVerif, injective events are declared without injective indices as the tool consider that all injective events have distinct injective indices. Internally, upon parsing the query, ProVerif associates a fresh injective index to each injective event and apply boolean distributivity rules to transform the query in disjunctive normal form. Lemma 1 guarantees the correctness of this transformation. ▶

Note that the secrecy of a closed term M can be expressed as the correspondence query $\text{att}_{\kappa}(M) \Rightarrow \bot$.

The previous definition gives us a generic definition of satisfiability of a correspondence query that mainly depends on a set of traces given as input. Of course, our main goal is to prove the the satisfiability of a correspondence query given some initial configuration within the semantics defined in fig. 3 and the relation $\vdash_o$. This is expressed in the following definition.

Definition 5. *Let* $\mathcal{C} = 0, \mathcal{E}, \mathcal{P}, \emptyset, \mathcal{A}$ *be a closed configuration such that* $names(\mathcal{A}) \cup names(\mathcal{P}) \subseteq \mathcal{E}$. *Let* ϱ *be a temporal correspondence query such that* $names(\varrho) \subseteq \mathcal{E}$. *We say that* $\mathcal{C}$ *satisfies* ϱ *when* $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)) \models \varrho$.

2.3.2 Equivalence properties

Privacy properties are often modeled using equivalence properties. In particular, observational equivalence, a weak bisimulation stable by context, intuitively guarantees that an attacker cannot see any difference between observationally equivalent processes. In [BAF08], it was shown that ProVerif can prove equivalence between two processes P and Q that differ only by the terms they contain. To do so, ProVerif represents P and Q as a single process (called biprocess). Then it proves observational equivalence between P and Q by proving a trace property on the biprocess. More specifically, they introduced a notion of convergent bitraces and showed that observational equivalence holds when all bitraces of the biprocess are convergent. In this paper, we will only focus on improving the verification of convergence of bitraces and we refer the reader to [BAF08] for the link between observational equivalence and biprocesses.

The grammar of biprocesses is the same as in Figure 2 with the addition of $\mathsf{diff}[M, M']$ for terms and $\mathsf{diff}[D, D']$ for expressions. Given a biprocess P, we define $\mathsf{fst}(P)$ (resp. $\mathsf{snd}(P)$) as the process obtained from P by replacing all instances of $\mathsf{diff}[M, M']$ with M (resp. M') and all instances of $\mathsf{diff}[D, D']$ with D (resp. D').

The semantics on biprocesses is defined by a relation $\rightarrow_{o'}$ obtained from the relation $\rightarrow_o$ from Figure 3 except that the rules (I/O), (Out), (In), (App) (Let1), (Let2), (Get1) and (Get2) are defined in Figure 4.

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{out}(N, M); P, \mathsf{in}(N', x); Q\}\!\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_{o'} \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \cup \{\!\!\{P, Q\{^M/_x\}\}\!\!\} \quad \text{(I/O)}$$
$$\text{if } \mathsf{fst}(N) = \mathsf{fst}(N') \text{ and } \mathsf{snd}(N) = \mathsf{snd}(N')$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{out}(N, M); P\}\!\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_{o'} \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{P\}\!\!\}, \mathcal{T}, \mathcal{A} \cup \{M\} \quad \text{(Out)}$$
$$\text{if } N' \in \mathcal{A}, \mathsf{fst}(N) = \mathsf{fst}(N') \text{ and } \mathsf{snd}(N) = \mathsf{snd}(N')$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{in}(N, x); Q\}\!\!\}, \mathcal{T}, \mathcal{A} \xrightarrow{\mathrm{msg}(N,M)}_{o'} \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{Q\{^M/_x\}\}\!\!\}, \mathcal{T}, \mathcal{A} \quad \text{(In)}$$
$$\text{if } N', M \in \mathcal{A}, \mathsf{fst}(N) = \mathsf{fst}(N'), \mathsf{snd}(N') = \mathsf{snd}(N')$$

$$\kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \rightarrow_{o'} \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A} \cup \{\mathsf{diff}[N_1, N_2]\} \quad \text{(App)}$$
$$\text{if } M_1, \ldots, M_n \in \mathcal{A},\ f/n \in \mathcal{F}_c \cup \mathcal{F}_d,$$
$$\mathsf{fst}(f(M_1, \ldots, M_n)) \Downarrow N_1 \text{ and } \mathsf{snd}(f(M_1, \ldots, M_n)) \Downarrow N_2$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}, \mathcal{T}, \mathcal{A} \rightarrow_{o'} \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{P\{^{\mathsf{diff}[M_1,M_2]}/_x\}\}\!\!\}, \mathcal{T}, \mathcal{A} \quad \text{(Let1)}$$
$$\text{if } \mathsf{fst}(D) \Downarrow M_1 \text{ and } \mathsf{snd}(D) \Downarrow M_2$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}, \mathcal{T}, \mathcal{A} \rightarrow_{o'} \kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{Q\}\!\!\}, \mathcal{T}, \mathcal{A} \quad \text{(Let2)}$$
$$\text{if } \mathsf{fst}(D) \Downarrow \mathsf{fail} \text{ and } \mathsf{snd}(D) \Downarrow \mathsf{fail}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}, \mathcal{T}, \mathcal{A} \rightarrow_{o'} \quad \text{(Get1)}$$
$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{P\sigma\}\!\!\}, \mathcal{T}, \mathcal{A}$$
$$\text{if } tbl(x_1, \ldots, x_n)\sigma \in \mathcal{T}, \mathsf{fst}(D\sigma) \Downarrow \mathit{true} \text{ and } \mathsf{snd}(D\sigma) \Downarrow \mathit{true}$$

$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}, \mathcal{T}, \mathcal{A} \rightarrow_{o'} \quad \text{(Get2)}$$
$$\kappa, \mathcal{E}, \mathcal{P} \cup \{\!\!\{Q\}\!\!\}, \mathcal{T}, \mathcal{A}$$
$$\text{if for all } \sigma,\ tbl(x_1, \ldots, x_n)\sigma \notin \mathcal{T} \text{ or } (\neg(\mathsf{fst}(D\sigma) \Downarrow \mathit{true}) \text{ and } \neg(\mathsf{snd}(D\sigma) \Downarrow \mathit{true}))$$

Figure 4: Semantics of biprocesses $\rightarrow_{o'}$

Note that biprocesses can be seen as a generalization of processes. Indeed, a process is a

biprocess P that does not contain any diff, i.e. $\mathsf{fst}(P) = \mathsf{snd}(P)$. Hence we can reuse every notion we defined on configurations and traces so far.

We introduce the notion of *convergent trace* that allows to prove observational equivalence.

Definition 6. *We say that $T \in \mathbb{T}(\rightarrow_{o'})$* converges*, denoted $T{\downarrow\uparrow}$, when for all steps τ, $T[\tau] = \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ implies:*

- *if $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{\mathsf{out}(N, M); P, \mathsf{in}(N', x); Q\}\!\!\}$ or $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{\mathsf{out}(N, M); P\}\!\!\}$ and $N' \in \mathcal{A}$ or $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{\mathsf{in}(N, x); P\}\!\!\}$, $N' \in \mathcal{A}$ then*

$$\mathsf{fst}(N) = \mathsf{fst}(N') \text{ iff } \mathsf{snd}(N) = \mathsf{snd}(N')$$

- *if $M_1, \ldots, M_n \in \mathcal{A}$, $f/n \in \mathcal{F}_c \cup \mathcal{F}_d$ then*

$$\mathsf{fst}(f(M_1, \ldots, M_n)) \Downarrow \mathsf{fail} \text{ iff } \mathsf{snd}(f(M_1, \ldots, M_n)) \Downarrow \mathsf{fail}$$

- *if $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}$ then $\mathsf{fst}(D) \Downarrow \mathsf{fail}$ iff $\mathsf{snd}(D) \Downarrow \mathsf{fail}$.*
- *if $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{\mathsf{get}\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\!\}$ and $tbl(x_1, \ldots, x_n)\sigma \in \mathcal{T}$ for some σ then $\mathsf{fst}(D\sigma) \Downarrow$ true iff $\mathsf{snd}(D\sigma) \Downarrow$ true*

We extend the notion to set of traces as expected, i.e. $S{\downarrow\uparrow}$ if and only if for all $T \in S$, $T{\downarrow\uparrow}$.

The following theorem shows the soundness of convergence w.r.t. observational equivalence.

Proposition 1 ([BAF08])**.** *For all closed biconfigurations $\mathcal{C}$, if $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'}){\downarrow\uparrow}$ then $\mathsf{fst}(\mathcal{C})$ and $\mathsf{snd}(\mathcal{C})$ are observationally equivalent (as defined in [BAF08]).*

2.3.3 Correspondence queries on bitraces

In this paper, we introduce the notion of correspondence queries on bitraces. Though the main goal of biprocesses is to prove equivalence queries, we use the correspondence queries on convergent bitraces to help the proof of equivalence by the means of axioms and lemmas (see section 2.4). Fundamentally, a correspondence query on convergent bitraces is exactly the same as a correspondence query on standard traces except that facts are replaced by bifacts. Formally, we consider the algebra for queries defined in Section 2.3.1 where injective events are removed and where we replace:

- events $\mathrm{event}(e(M_1, \ldots, M_n))$ by $\mathrm{event}'(e(M_1, \ldots, M_n), e(M'_1, \ldots, M'_n))$;
- facts $\mathrm{att}_\kappa(M)$ by $\mathrm{att}'_\kappa(M, M')$
- facts $\mathrm{msg}_\kappa(M, N)$ by $\mathrm{msg}'_\kappa(M, N, M', N')$
- facts $\mathrm{table}_\kappa(tbl(M_1, \ldots, M_n))$ by $\mathrm{table}'_\kappa(tbl(M_1, \ldots, M_n), tbl(M'_1, \ldots, M'_n))$

We can also define the satisfaction relation $\vdash_{o'}$ similarly to $\vdash_o$. For instance, given a bitrace T and a step τ, $T, \tau \vdash_{o'} \mathrm{att}'_\kappa(M, M')$ if and only if $T[\tau] \rightarrow^*_{o'} \kappa, \mathcal{E}, \mathcal{P}, \mathcal{T}, \mathcal{A}$ by only the rules App, New and Phase such that there exists $M'' \in \mathcal{A}$ with $\mathsf{fst}(M'') = M$ and $\mathsf{snd}(M'') = M'$.

Definition 7. *Let $\mathcal{C} = 0, \mathcal{E}, \mathcal{P}, \emptyset, \mathcal{A}$ be a closed biconfiguration such that $names(\mathcal{A}) \cup names(\mathcal{P}) \subseteq \mathcal{E}$. Let ϱ be temporal a correspondence query on bitraces such that $names(\varrho) \subseteq \mathcal{E}$. We say that $\mathcal{C}$ satisfies ϱ when $(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \rightarrow_{o'})) \models \varrho$.*

2.4 Axioms, restriction and lemmas

In order to verify correspondence and equivalence queries, ProVerif intuitively transforms the process into a set of Horn clauses and applies a saturation procedure based on resolution. Once a fixpoint is reached, the algorithm will verify the query only on the saturated clauses. Note that the clauses generated and the saturation procedure for the correspondence queries and for equivalence queries differ but the main idea of the algorithms stays the same: (i) generate initial set of Horn clauses (ii) saturate the set of clauses until a fixpoint is reached (iii) verify the query on the saturated set of Horn clauses. When ProVerif is not able to terminate (more precisely, when it does not seem to terminate), it is generally due to the saturation procedure which does not reach a fixpoint. To help ProVerif terminate, several heuristics are available (modifying for instance how names are represented internally or on which clauses the resolution is applied). In this paper, we will focus only on the *secrecy heuristic* (details on all available heuristics can be found in [Bla16, BSCS17]).

In a ProVerif input file, a user is allowed to declare terms, say M, as secret, meaning that the input process should satisfy the secrecy of any ground instances of M. These declarations can be seen as *intermediate properties* for the saturation procedure. Indeed, ProVerif can remove during the saturation procedure Horn clauses that contradict the secrecy of M. Since fewer clauses are generated, this potentially helps the saturation procedure to terminate. Of course, ProVerif also verifies that the secrecy of M is really satisfied by the protocol ensuring that ProVerif never proves something incorrect.

In this paper, we generalize these intermediate properties, called *lemmas*, to a subclass of correspondence queries. Typically, ProVerif will first prove theses lemmas and then use them during the saturation procedure of the main query. We also allow the declaration of *axioms* that are properties that are true but do not need to be proved. Finally, we introduce *restrictions* to model assumptions: only traces that satisfy the restrictions will be considered. It can be used *e.g.* to specify that a ressource can be accessed by at most one process, without a heavy encoding with private channels.

Note that this idea of lemmas and axioms is not novel in the field of automatic verifier: A proof in Tamarin usually consists of many lemmas and axioms; and a proved lemma or an axiom can be used to help proving other lemmas along the road.

Definition 8. *A* ProVerif lemma, axiom, restriction *is a atemporal correspondence query* $F_1 \wedge \ldots \wedge F_n \Rightarrow \psi$ *such that* ψ *only contain events and generic predicates (i.e. no injective event, nested query, attacker fact, message fact and table fact).*

Example 9. The ProVerif lemma $\mathrm{att}(M) \Rightarrow \bot$ corresponds to a secrecy declaration in ProVerif 2.00. ▶

As previously mentioned, restrictions model assumptions meaning that given a query ϱ and a set of restrictions $\mathcal{R}$, we will be interested to prove the query ϱ only on the trace that are satisfied by all restrictions in $\mathcal{R}$. Formally, given $\mathcal{C} = 0, \mathcal{E}, \mathcal{P}, \emptyset, \mathcal{A}$ a closed configuration such that $\mathit{names}(\mathcal{A}) \cup \mathit{names}(\mathcal{P}) \subseteq \mathcal{E}$, given ϱ be a temporal correspondence query such that $\mathit{names}(\varrho) \subseteq \mathcal{E}$, given $\mathcal{R}$ a set of restrictions, we want to prove the following:

$$(\vdash_o, \{T \in \mathsf{trace}(\mathcal{C}, \rightarrow_o) \mid \forall \varrho' \in \mathcal{R}, (\vdash_o, \{T\}) \models \varrho'\} \models \varrho$$

To simplify the reading, we will denote the above formula as follows: $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)_{|\mathcal{R}}) \models \varrho$.

3 Instrumented processes

ProVerif generates a set of clauses representing both the protocol and the attacker capabilities. Intuitively, these clauses mimic the semantics of processes. However, the freshness requirement of the rule Restr is difficult to encode in clauses. To handle this, [BAF08, Bla09] introduce instrumented processes in which replications are indexed with some identifier and all names a are replaced by a pattern $a[x_1, \ldots, x_n]$ where the variables $x_1, \ldots, x_n$ model which input and replication this name depends on. Therefore, two names depending on different replication identifiers will be considered as different, i.e. thus modeling the freshness of the term.

Syntax. ProVerif models messages inside clauses by patterns and may-fail patterns defined as follows:

$pt ::=$	pattern
x, y, z	variables
i	variable session identifier
λ	constant session identifier
$f(pt_1, \ldots, pt_n)$	constructor application
$a[pt_1, \ldots, pt_n]$	name
$mpt ::=$	may-fail pattern
u, v	may-fail variables
pt	pattern
fail	failure

An instrumented process is intuitively a process where each occurence of a replication, input, event, or table lookup in a process is associated with an occurrence label in the form of a name usually denoted o. We require that, in the initial process, each of these names occurs at most once.

Example 10. Consider a process $P = !\mathsf{in}(c, x); \mathsf{out}(c, h(x)); \mathsf{in}(c, y)$. A corresponding instrumented process of P would be the process $!^{o_1}\ \mathsf{in}^{o_2}(c, x); \mathsf{out}(c, h(x)); \mathsf{in}^{o_3}(c, y)$. ▶

During the saturation procedure, we will need to distinguish the standard variables that are used in the protocol and the variables for session identifier. Hence, we consider a new infinite set of variables denoted $\mathcal{V}_!$ and distinct from $\mathcal{V}$. Similarly, we consider a new infinite set of constant session identifier, denoted Λ.

Semantics. An instrumented configuration $\kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ is similar to a configuration with the following differences:

- Λ is the set of already used session identifiers
- ρ is a mapping from names to patterns
- $\mathcal{P}$ is a multiset of quadruples $(P, \mathcal{O}, \mathcal{I})$ where:
  - P is an instrumented process

– $\mathcal{O}$ is a list of names corresponding to the occurence labels of already reduced inputs, reduced table lookup and replications that occur above P

– $\mathcal{I}$ is the list of patterns corresponding to terms received by the inputs, lookup tables and of session identifiers of replications that occur above P.

The semantics is displayed in Figure 5. We say that an initial instrumented configuration is a closed instrumented configuration of the form $0, \rho, \{\!\{(P, \emptyset, \emptyset)\}\!\}, \emptyset, \mathcal{A}, \emptyset$ where $\mathcal{A}$ only contains names, $dom(\rho)$ is the union of $\mathcal{A}$ with the free names of P and $a\rho = a[]$ for all $a \in dom(\rho)$. We will denote such initial instrumented configuration $\rho, P, \mathcal{A}$.

The initial instrumented configuration associated to the initial configuration $\mathcal{E}, P, \mathcal{A}$ is the configuration $\rho, P, \mathcal{A}$ where $\rho = \{a \mapsto a[] \mid a \in \mathcal{E}\}$.

We can show the links between the different sequences of labels and patterns $(\mathcal{O}, \mathcal{I})$ found in the instrumented semantics.

Definition 9. *Let* $(\mathcal{O}_1, \mathcal{I}_1) = ([o_1, \ldots, o_n], [pt_1, \ldots, pt_n])$ *and* $(\mathcal{O}_2, \mathcal{I}_2) = ([o'_1, \ldots, o'_m], [pt'_1, \ldots, pt'_m])$ *two pairs of sequences of occurrence labels and sequences of patterns. We say that* $(\mathcal{O}_1, \mathcal{I}_1)$ *and* $(\mathcal{O}_2, \mathcal{I}_2)$ *are* compatible *when for all* $k \in \mathbb{N}$*, for all* $i \leq k$*, if the* i*th pattern in* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *coincides and is a session identifier, that is* $pt_i = pt'_i$*,* pt_i *is a session identifier, and if the labels coincide in* $\mathcal{O}_1$ *and* $\mathcal{O}_2$ *up to the index* k*, that is for all* $j \leq k$*,* $o_j = o'_j$ *then*

- *the previous patterns coincide as well: for all* $\ell \leq i$*,* $pt_\ell = pt'_\ell$*;*
- *and the patterns still coincide until the next session identifier: for all* $k \geq \ell > i$*, if for all* $i < \ell' \leq \ell$*,* $pt_{\ell'}$ *is not a session identifier then* $pt_\ell = pt'_\ell$*.*

Sequences of labels and patterns $(\mathcal{O}, \mathcal{I})$ found in successive configurations are compatible.

Lemma 2. *Let* $\mathcal{C}$ *be a closed instrumented configuration. For all* $\mathcal{C} \to_i^* n_1, \rho_1, \mathcal{P}_1, \mathcal{T}_1, \mathcal{A}_1, \Lambda_1 \to_i^* n_2, \rho_2, \mathcal{P}_2, \mathcal{T}_2, \mathcal{A}_2, \Lambda_2$*, for all* $(P_1, \mathcal{O}_1, \mathcal{I}_1) \in \mathcal{P}_1$*, for all* $(P_2, \mathcal{O}_2, \mathcal{I}_2) \in \mathcal{P}_2$*,* $(\mathcal{O}_1, \mathcal{I}_1)$ *and* $(\mathcal{O}_2, \mathcal{I}_2)$ *are compatible.*

This lemma is intuitively a consequence of the application of the rule I-Repl. More specifically, notice that every time this rule is applied, we instantiate a replication variables by a session identifier not already used, i.e. that is not in Λ. Therefore, if two instances of an instrumented name $a[pt_1, \ldots, p_n]$ and $a[pt'_1, \ldots, pt'_n]$ can be found in a trace with $pt_i = pt'_i \in \Lambda$, we can deduce that all the instantiations of replication variables should be the same. Moreover, this property also expresses the fact that each input can only be executed once within a single trace.

It remains to show the soundness of our transformation. The instrumented semantics introduce patterns that allows us to encode the freshness of names within patterns. Thus, in a query, we consider atomic formula that contains patterns instead of terms. To simplify the reading, we overload the predicates att_κ, msg_κ, ... to take as arguments patterns. Moreover, we also overload the predicate event as being a predicate of arity 2 that includes the occurrence pattern (similarly for the predicate inj_k-event. Thus, when ev is an event pattern and o is an occurrence pattern, we write $\mathrm{event}(o, ev)$ for the predicate that describe the fact that the event ev was trigger at occurrence o. Adding the occurrence as additional argument will be useful later on for proving injective queries.

Formally, the satisfaction relation $\vdash_i$ on traces of $\mathbb{T}(\to_i)$ is defined by adapting the satisfaction relation $\vdash_o$ to instrumented configurations. For instance, $T, \tau \vdash_i \mathrm{event}(o, ev)$ (resp.

$$\mathcal{P} \cup \{\!\{(0, \mathcal{O}, \mathcal{I})\}\!\} \to_i \mathcal{P} \quad \text{(I-Nil)}$$

$$\mathcal{P} \cup \{\!\{(P \mid Q, \mathcal{O}, \mathcal{I})\}\!\} \to_i \mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I}), (Q, \mathcal{O}, \mathcal{I})\}\!\} \quad \text{(I-Par)}$$

$$\mathcal{P} \cup \{\!\{(!^o P, \mathcal{O}, \mathcal{I})\}\!\}, \Lambda \to_i \mathcal{P} \cup \{\!\{(P, (\mathcal{O}, o), (\mathcal{I}, \lambda)), (!^o P, \mathcal{O}, \mathcal{I})\}\!\}, \Lambda \cup \{\lambda\} \quad \text{if } \lambda \notin \Lambda \quad \text{(I-Repl)}$$

$$\rho, \mathcal{P} \cup \{\!\{(\mathsf{new}\ a; P, \mathcal{O}, \mathcal{I})\}\!\} \to_i (\rho[a' \mapsto a[\mathcal{I}]]), \mathcal{P} \cup \{\!\{(P\{^{a'}/_{a}\}, \mathcal{O}, \mathcal{I})\}\!\} \quad \text{(I-Restr)}$$
$$\text{where } a' \notin dom(\rho)$$

$$\kappa, \rho, \mathcal{P} \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}, \mathcal{I}), (\mathsf{in}^o(N, x); Q, \mathcal{O}', \mathcal{I}')\}\!\} \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i \quad \text{(I-I/O)}$$
$$\kappa, \rho, \mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I}), (Q\{^{M}/_{x}\}, (\mathcal{O}', o), (\mathcal{I}', M\rho))\}\!\}$$

$$\kappa, \rho, \mathcal{A} \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i \kappa, \rho, \mathcal{A} \quad \text{if } N, M \in \mathcal{A} \quad \text{(I-Msg)}$$

$$\mathcal{P} \cup \{\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\} \to_i \mathcal{P} \cup \{\!\{(P\{^{M}/_{x}\}, \mathcal{O}, \mathcal{I})\}\!\} \quad \text{if } D \Downarrow M \quad \text{(I-Let1)}$$

$$\mathcal{P} \cup \{\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\} \to_i \mathcal{P} \cup \{\!\{(Q, \mathcal{O}, \mathcal{I})\}\!\} \quad \text{if } D \Downarrow \mathsf{fail} \quad \text{(I-Let2)}$$

$$\mathcal{P} \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{A} \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i \mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{A} \cup \{M\} \quad \text{if } N \in \mathcal{A} \quad \text{(I-Out)}$$

$$\kappa, \rho, \mathcal{P} \cup \{\!\{(\mathsf{in}^o(N, x); Q, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{A} \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i \quad \text{(I-In)}$$
$$\kappa, \rho, \mathcal{P} \cup \{\!\{(Q\{^{M}/_{x}\}, (\mathcal{O}, o), (\mathcal{I}, M\rho))\}\!\}, \mathcal{A}$$
$$\text{if } N, M \in \mathcal{A}$$

$$\mathcal{P}, \mathcal{A} \to_i \mathcal{P}, \mathcal{A} \cup \{M\} \quad \text{(I-App)}$$
$$\text{if } M_1, \ldots, M_n \in \mathcal{A},\ f/n \in \mathcal{F}_c \cup \mathcal{F}_d \text{ and } f(M_1, \ldots, M_n) \Downarrow M$$

$$\rho, \mathcal{A} \to_i (\rho[a' \mapsto b_0[\lambda]]), \mathcal{A} \cup \{a'\} \quad \text{if } a' \notin dom(\rho) \text{ and } b_0[\lambda] \notin img(\rho) \quad \text{(I-New)}$$

$$\rho, \mathcal{P} \cup \{\!\{(\mathsf{event}^o(ev); P, \mathcal{O}, \mathcal{I})\}\!\} \xrightarrow{\mathrm{event}(o[\mathcal{I}], ev\rho)}_i \rho, \mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\} \quad \text{(I-Event)}$$

$$\kappa, \mathcal{P} \cup \mathcal{P}_{>\kappa+1} \cup \{\!\{(\mathsf{phase}\ \kappa+1; P_i, \mathcal{O}_i, \mathcal{I}_i)\}\!\}_{i=1}^{k} \to_i \kappa+1, \mathcal{P}_{>\kappa+1} \cup \{\!\{(P_i, \mathcal{O}_i, \mathcal{I}_i)\}\!\}_{i=1}^{k} \quad \text{(I-Phase)}$$
$$\text{if all processes of } \mathcal{P} \text{ do not start with some } \mathsf{phase}\ \kappa',\ \kappa' > \kappa$$
$$\text{and all processes of } \mathcal{P}_{>\kappa+1} \text{ start with some } \mathsf{phase}\ \kappa',\ \kappa' > \kappa+1$$

$$\mathcal{P} \cup \{\!\{(\mathsf{insert}\ tbl(M_1, \ldots, M_n); P, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{T} \to_i \quad \text{(I-Insert)}$$
$$\mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{T} \cup \{tbl(M_1, \ldots, M_n)\}$$

$$\kappa, \rho, \mathcal{P} \cup \{\!\{(\mathsf{get}^o\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{T} \to_i \quad \text{(I-Get1)}$$
$$\kappa, \rho, \mathcal{P} \cup \{\!\{(P\sigma, (\mathcal{O}, o), (\mathcal{I}, tbl(x_1, \ldots, x_n)\sigma\rho))\}\!\}, \mathcal{T}$$
$$\text{if } tbl(x_1, \ldots, x_n)\sigma \in \mathcal{T} \text{ and } D\sigma \Downarrow true$$

$$\mathcal{P} \cup \{\!\{(\mathsf{get}^o\ tbl(x_1, \ldots, x_n)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{T} \to_i \quad \text{(I-Get2)}$$
$$\mathcal{P} \cup \{\!\{(Q, \mathcal{O}, \mathcal{I})\}\!\}, \mathcal{T}$$
$$\text{if for all } \sigma,\ tbl(x_1, \ldots, x_n)\sigma \notin \mathcal{T} \text{ or } \neg(D\sigma \Downarrow true)$$

For clarity sake, we only show the components of the configuration that are either modified or used by the rules.

Figure 5: Instrumented semantics $\to_i$

$\text{inj}_k\text{-event}(o, ev)$) if and only if $T[\tau - 1] \xrightarrow{\text{event}(o,ev)}_i T[\tau]$. For the attacker predicate, we apply the mapping ρ that transforms names into patterns: $T, \tau \vdash_i \text{att}_\kappa(M\rho)$ if and only if $T[\tau] \rightarrow_i^* \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ by only the rules I-App, I-New and I-Phase such that $M \in \mathcal{A}$. Similarly, $T, \tau \vdash_i \text{msg}_\kappa(N, M)$ if and only if $T[\tau - 1] \xrightarrow{\text{msg}(N,M)}_i T[\tau]$ with $T[\tau] = \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$. Note that for the predicate $\text{msg}_\kappa(N, M)$, N and M are already patterns by definition of the instrumented semantics (see Figure 5).

Finally, to show the soundness of our transformation, we need to show how we transform a query into a query with patterns. Given $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ an initial configuration, a query ϱ must always satisfy $names(\varrho) \subseteq \mathcal{E}$. Thus, the basic transformation will be to apply the mapping ρ on ϱ where $\rho = [a \mapsto a[] \mid a \in names(\varrho)]$. However, event pattern predicates take an additional argument (the occurrence). Therefore, we modify the query ϱ by replacing all atomic formulae α in ϱ with: (i) $\alpha\rho$ when α is not an event fact; and (ii) $\text{event}(x, ev\rho)$ with x a fresh variable when $\alpha = \text{event}(ev)$. We denote by $[\varrho]_i$ the query obtained by this transformation.

Lemma 3. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial configuration and let $\mathcal{C}_I$ be its associated initial instrumented configuration. Let ϱ be a correspondence query such that $names(\varrho) \subseteq \mathcal{E}$. Let $\mathcal{R}$ be a set of restrictions.*

We have $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)_{|\mathcal{R}}) \models \varrho$ if and only if $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)_{|[\mathcal{R}]_i}) \models [\varrho]_i$.

For equivalences properties, similarly to how we defined semantics for biprocess from $\rightarrow_o$, we also define an instrumented semantics $\rightarrow_{i'}$ for biprocess from $\rightarrow_i$. In a query for bitrace, the event $\text{event}(ev)$ is replaced by the event $\text{event}'(ev_1, ev_2)$ that already takes two arguments (the value of the event on the first and second projection of the trace). Though we overloaded the predicate event with the occurrence for the instrumented semantics, we do not consider additional arguments for the event pattern predicate for bitrace in the instrumented semantics. Hence, $\text{event}'(ev_1, ev_2)$ will only be replaced by $\text{event}'(ev_1\rho, ev_2\rho)$. As we explained earlier, we introduce occurrence as additional argument to help proving injective queries. However, correspondence queries on bitrace will only be used as lemmas or restrictions to help proving an equivalence query. Since lemmas are non-injective queries (see Definition 8), we can therefore omit the occurrence in the events for bitraces. Thus, given a query for bitrace ϱ, we denote $[\varrho]_{i'}$ the query with patterns obtained from ϱ by replacing all atomic formulae α in ϱ by $\alpha\rho$ where $\rho = [a \mapsto a[] \mid a \in names(\varrho)]$. Finally, we define the satisfaction relation $\vdash_{i'}$ from $\vdash_i$ and the convergence of trace $T \in \mathbb{T}(\rightarrow_{i'})$, denoted $T{\Downarrow\Uparrow}_i$, by adapting the convergence $\Downarrow\Uparrow$ to instrumented traces as expected.

Lemma 4. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial biconfiguration and let $\mathcal{C}_I$ be its associated initial instrumented biconfiguration. Let $\mathcal{R}$ be a set of bi-restrictions. We have $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}{\Downarrow\Uparrow}$ if and only if $\mathsf{trace}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}{\Downarrow\Uparrow}$.*

Lemma 5. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial biconfiguration and let $\mathcal{C}_I$ be its associated initial instrumented biconfiguration. Let ϱ be a correspondence query on bitraces such that $names(\varrho) \subseteq \mathcal{E}$. Let $\mathcal{R}$ be a set of bi-restrictions. We have:*

$$(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}) \models \varrho \text{ if and only if } (\vdash_{i'}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}) \models [\varrho]_{i'}$$

3.1 Transforming temporal queries in atemporal queries

The core algorithms in ProVerif can only prove atemporal correspondence queries. We show in this section how we can encode temporal queries into atemporal ones, mostly relying on

nested queries and occurrences. We first link in the following lemma occurrence and steps in the trace.

Lemma 6. *Let $\mathcal{C}$ be an initial instrumented configuration. Let $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. For all ground facts $F = \mathrm{event}(o, ev), F' = \mathrm{event}(o', ev')$, for all steps τ, τ', if $T, \tau \vdash_i F$ and $T, \tau' \vdash_i F'$ then $\tau = \tau'$ if and only if $o = o'$.*

Proof. The right implication is trivial since two different events cannot be satisfied at the same step. For the left implication, we first need to recall that in the initial instrumented configuration, each occurrence can occurs at most once. After some transition it is possible that an occurrence name occurs multiple times due to the replication rule. However, each application of the rule I-Repl on $\mathcal{P} \cup \{\!\{(!^o P, \mathcal{O}, \mathcal{I})\}\!\}, \Lambda$ generates a fresh session identifier that is added in $\mathcal{I}$, that is $(P, (\mathcal{O}, o), (\mathcal{I}, \lambda))$ with $\lambda \notin \Lambda$. Finally, only the rule I-Event, i.e. $\rho, \mathcal{P} \cup \{\!\{(\mathsf{event}^o(ev); P, \mathcal{O}, \mathcal{I})\}\!\} \xrightarrow{\mathrm{event}(o[\mathcal{I}], ev\rho)}_i \rho, \mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$, can emit events. Hence, if $T, \tau \vdash_i F$ with $F = \mathrm{event}(o, ev)$ being a ground fact, we necessarily have that o is a name of the form $o'[\mathcal{I}']$ in which $\mathcal{I}'$ includes all session identifiers. Since all session identifiers are always freshly generated and the occurrence names occur at most once in the initial instrumented configuration, we conclude that two events satisfied at different steps of the trace necessarily either have different occurrence names or different session identifiers. □

Thanks to Lemma 6, we can now express disequalities and equalities between temporal variables as disequalities and equalities between occurrences. Inequalities between temporal variables will be represented as nested queries.

Consider an initial configuration $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ and a temporal query ϱ such that $names(\varrho) \subseteq \mathcal{E}$. Consider $\rho = [a \mapsto a[] \mid a \in names(\varrho)]$. The transformation of ϱ to an atemporal query, denoted $[\varrho]_i^{@}$, has some restrictions as some queries cannot be transformed into atemporal query.

Formally, $[\varrho]_i^{@}$ raises an error when there exists two temporal variables i, j bound respectively by $F@i, G@j$ in ϱ and there exists:

- $i = j$ or $i \neq j$ in ϱ such that either F or G is not an event.
- $i > j$ or $i \geq j$ in ϱ such that F is not an event.

Assuming that $[\varrho]_i^{@}$ does not raises an error, the transformation proceeds as follows:

1. For all temporal variables i and j bound respectively by $F@i$ and $G@j$ in ϱ, replace all occurrences of $i > j$ in ϱ by $F@i \rightsquigarrow G@j \wedge j \neq i$ if G is an event else $F@i \rightsquigarrow G@j$
2. For all temporal variables i and j bound respectively by $F@i$ and $G@j$ in ϱ, replace all occurrences of $i \geq j$ in ϱ by $F@i \rightsquigarrow G@j$.
3. Generate a bijection β from the temporal variables in ϱ to fresh variables.
4. For all temporal variables i and j bound respectively by $F@i$ and $G@j$ in ϱ, replace all occurrences of $i = j$ (resp. $i \neq j$) in ϱ by $i\beta = j\beta$ (resp. $i\beta \neq j\beta$).
5. Replace all occurrences of
   (a) $\mathrm{event}(ev)@i$ (resp. $\mathrm{inj}_k\text{-event}(ev)@i$) by $\mathrm{event}(i\beta, ev\rho)$ (resp. $\mathrm{inj}_k\text{-event}(i\beta, ev\rho)$).

(b) event(ev) (resp. inj$_k$-event(ev)) by event($x, ev\rho$) (resp. inj$_k$-event($x, ev\rho$)) where x is a fresh variable.

(c) facts $F[@i]$ where F is not an (injective) event by $F\rho$

(d) generic formulas ϕ by $\phi\rho$.

Lemma 7. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial configuration and let $\mathcal{C}_I$ be its associated initial instrumented configuration. Let ϱ be a temporal correspondence query such that names$(\varrho) \subseteq \mathcal{E}$. let $\mathcal{R}$ be a set of restrictions. If $[\varrho]_i^{@}$ does not raise an error then we have $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)_{|\mathcal{R}}) \models \varrho$ if and only if $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)_{|[\mathcal{R}]_i}) \models [\varrho]_i^{@}$.*

Since we showed how to transform a temporal correspondence query into an atemporal correspondence query, for the rest of the paper, we will always assume that correspondence query are atemporal.

3.2 Restricting the trace search space

We restrict the traces we consider in the instrumented semantics in order to speed up the saturation procedure. First, we want to restrict the tuples $(\tau_1, \ldots, \tau_n)$ we verify when proving $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)) \models \varrho$. In particular, in Definition 4, when we consider a tuple $(\tau_1, \ldots, \tau_n)$ and a substitution σ, we check that $T, \tau_j \vdash_i F_j\sigma$. However, when F_j is an attacker or table fact, if $T, \tau_j \vdash_i F_j\sigma$ then $T, \tau_j' \vdash F_j\sigma$ for all $\tau_j' \geq \tau_j$. Thus, instead of considering all the steps where $F_j\sigma$ holds, we will intuitively consider only the first step on which $F_j\sigma$ holds. Hence, we need to restrict the satisfaction relation $\vdash_i$ for the premise of the query. Note that this restriction in the search space concerns only the satisfaction relation for the premise of the query. To preserve soundness, we do not want to restrict the satisfaction relation for the conclusion of the query. As such, we extend Definition 4 by distinguishing these satisfaction relations:

Definition 10. *Let $\varrho = (\bigwedge_{i=1}^n F_i \Rightarrow \psi)$ be a correspondence query. Let $\rightarrow$ be a binary relation.*

For all satisfaction relations $\vdash_\forall$, $\vdash_\exists$ on ground atomic formulas for $\rightarrow$, for all $\mathcal{S} \subseteq \mathbb{T}(\rightarrow)$, we say that $\vdash_\forall, \vdash_\exists, S$ satisfy ϱ, denoted $(\vdash_\forall, \vdash_\exists, \mathcal{S}) \models \varrho$ if and only if for all $T \in \mathcal{S}$, there exists an annotated query conclusion Ψ w.r.t. (n, m) such that $m = \mathsf{max}_{step}(T)$, $\overline{\Psi} = \psi$ and:

- *for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, for all substitutions σ, if $T, \tau_j \vdash_\forall F_j\sigma$ for $j = 1 \ldots n$ then there exists σ' such that $F_j\sigma = F_j\sigma'$ for $j = 1 \ldots n$ and $(\vdash_\exists, T, \tilde{\tau}) \models \Psi\sigma'$.*
- *for all* inj$_k$-event$(o, ev)^\mu$ *occurring in Ψ, $\mu(\tilde{\tau}, \sigma) = \mu(\tilde{\tau}', \sigma')$ implies that $\tau_j = \tau_j'$ for all j such that $F_j =$* inj$_{k_j}$-event(o_j, ev_j) *for some k_j, ev_j.*
- *for all* inj$_{k_1}$-event$(o_1, ev_1)^{\mu_1}$ *and* inj$_{k_2}$-event$(o_2, ev_2)^{\mu_2}$ *occurring in Ψ, $k_1 = k_2$ implies $\mu_1 = \mu_2$*

When verifying a query, we will often call $\vdash_\forall$ the universal satisfaction relation *and $\vdash_\exists$ the* existential satisfaction relation*.*

Given a set of restrictions $\mathcal{R}$, we will denote $(\vdash_\forall, \vdash_\exists, \mathcal{S}_{|\mathcal{R}}) \models \varrho$ for the following:

$$(\vdash_\forall, \vdash_\exists, \{T \in \mathcal{S} \mid \forall \varrho' \in \mathcal{R}, (\vdash_\forall, \vdash_\exists, \{T\}) \models \varrho'\}) \models \varrho$$

As mentioned before, the existential satisfaction relation will still be the relation $\vdash_i$ on instrumented traces. Hence, it remains to define the universal satisfaction relation that we will use to restrict the search space. Moreover, we will also restrict the set of traces we consider by only taking a subset of $\mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. The rest of this section is dedicated to the definition of this subset and the universal satisfaction relation we will consider.

3.2.1 Data constructor function symbols

A data constructor function is always associated with all its projection symbols. Thus, when an attacker learns a term $M = f(M_1, \ldots, M_n)$, he can always retrieve $M_1, \ldots, M_n$ by applying the different projections of f to M. This property is reflected in ProVerif during the saturation and verification procedure: a predicate $\mathrm{att}_\kappa(f(M_1, \ldots, M_n))$ in the hypothesis of a clause will be automatically transformed in the conjunction $\mathrm{att}_\kappa(M_1) \wedge \ldots \wedge \mathrm{att}_\kappa(M_n)$. A similar transformation occurs when $\mathrm{att}_\kappa(f(M_1, \ldots, M_n))$ is the conclusion of a clause, i.e. $H \rightarrow \mathrm{att}_\kappa(f(M_1, \ldots, M_n))$. In such a case, the clause will be transformed in n clauses $H \rightarrow \mathrm{att}_\kappa(M_1)$, ..., $H \rightarrow \mathrm{att}_\kappa(M_n)$. In term of traces, this transformation intuitively corresponds to forcing the attacker to always decompose terms with data constructor function symbols as root (to obtain the projection) and then recomposing the term itself (so that $f(M_1, \ldots, M_n)$ is built from $M_1, \ldots, M_n$). To describe these traces, we define three relations $\overset{M}{\Longrightarrow}_{\mathsf{dr}}$, $\overset{M}{\Longrightarrow}_{\mathsf{d},k}$ and $\overset{M}{\Longrightarrow}_{\mathsf{r}}$ on configurations that describe the deconstruction and reconstruction of terms with data constructor function symbols. Intuitively, the relation $\overset{M}{\Longrightarrow}_{\mathsf{r}}$ applies constructor symbols, the relation $\overset{M}{\Longrightarrow}_{\mathsf{d},k}$ deconstructs data always following the k-th component, while $\overset{M}{\Longrightarrow}_{\mathsf{dr}}$ corresponds to reconstructing data once they have been deconstructed. To ease the reading, when $T[\tau] \rightarrow_i T[\tau+1]$ by application of the rule I-App with the function f/n on terms $M_1, \ldots, M_n$, we will write $T[\tau] \xrightarrow{\text{I-App}(f, M_1, \ldots, M_n)}_i T[\tau+1]$.

Moreover, we denote $\mathsf{data}(T, \tau)$ the set of terms that have already been deconstructed / reconstructed up to the step τ. Formally, if κ is the phase of $T[\tau]$ then $M \in \mathsf{data}(T, \tau)$ if and only if there exists $\tau' \leq \tau$ and $T[\tau'] = \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ such that $M \in \mathcal{A}$ and if $M = f(M_1, \ldots, M_m)$ and $f \in \mathcal{F}_{data}$ then $T[\tau'-1] \xrightarrow{\text{I-App}(f, M_1, \ldots, M_m)}_i T[\tau']$.

Definition 11. *We define the relations $\overset{M}{\Longrightarrow}_{\mathsf{dr}}$, $\overset{M}{\Longrightarrow}_{\mathsf{d},k}$ and $\overset{M}{\Longrightarrow}_{\mathsf{r}}$ on configurations as follows: For all $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$, for all $f/m \in \mathcal{F}_{data}$, for all terms $M, M_1, \ldots, M_m$*

Reconstruction relation $\overset{M}{\Longrightarrow}_{\mathsf{r}}$:

$$T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{r}} T[\tau] \quad \text{if } M \in \mathsf{data}(T, \tau)$$

$$T[\tau] \overset{f(M_1, \ldots, M_m)}{\Longrightarrow}_{\mathsf{r}} T[\tau+1] \quad \text{if } M_1, \ldots, M_m \in \mathsf{data}(T, \tau) \text{ and } T[\tau] \xrightarrow{\text{I-App}(f, M_1, \ldots, M_m)}_i T[\tau+1]$$

Deconstruction relation $\overset{M}{\Longrightarrow}_{\mathsf{d},k}$:

$$T[\tau] \overset{f(M_1, \ldots, M_m)}{\Longrightarrow}_{\mathsf{d},k} T[\tau] \qquad \text{if } M_k \in \mathsf{data}(T, \tau)$$

$$T[\tau] \overset{f(M_1, \ldots, M_m)}{\Longrightarrow}_{\mathsf{d},k} T[\tau'] \qquad \text{if } T[\tau] \xrightarrow{\text{I-App}(\pi_k^f, f(M_1, \ldots, M_m))}_i T[\tau+1] \overset{M_k}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$$

Reconstruction/deconstruction relation $\overset{M}{\Longrightarrow}_{\mathsf{dr}}$:

$$T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{dr}} T[\tau] \qquad \textit{if } M \in \mathsf{data}(T,\tau)$$

$$T[\tau] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{dr}} T[\tau_m + 1]$$
$$\textit{if } T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{d},1} T[\tau_1] \overset{M}{\Longrightarrow}_{\mathsf{d},2} \ldots \overset{M}{\Longrightarrow}_{\mathsf{d},m} T[\tau_m] \overset{M}{\Longrightarrow}_{\mathsf{r}} T[\tau_m + 1] \textit{ with } M = f(M_1,\ldots,M_m)$$

Intuitively, the relation $\overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{r}}$ represents the reconstruction of $f(M_1,\ldots,M_m)$ by application of the rule I-App with f on $M_1,\ldots,M_m$ whereas the relations $\overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},k}$ represents the deconstruction of $f(M_1,\ldots,M_m)$ by application of the k-th projection of f. The main relation $T[\tau] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$ indicates that between the step τ and τ', the trace first deconstructs the term $f(M_1,\ldots,M_m)$ (i.e. $T[\tau_0] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},1} T[\tau_1] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},2} \ldots \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},m} T[\tau_m]$) and then reconstructs it (i.e. $T[\tau_m] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{r}} T[\tau_m + 1]$).

Note that a configuration $T[\tau]$ is in relation with itself w.r.t. to the relation $\overset{M}{\Longrightarrow}_{\mathsf{r}}$ and $\overset{M}{\Longrightarrow}_{\mathsf{dr}}$ when M have already been reconstructed, i.e. $M \in \mathsf{data}(T,\tau)$. A similar property holds for the relation $\overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},k}$ but focusing on the k-th projection, i.e. $M_k \in \mathsf{data}(T,\tau)$.

Finally, note that the relation $\overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},k}$ and $\overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{dr}}$ are mutually recursive in order to handle terms with nested data constructor symbols. For instance, when M_1 also contains a data constructor symbol as root then $T[\tau] \overset{f(M_1,\ldots,M_m)}{\Longrightarrow}_{\mathsf{d},1} T[\tau']$ indicates that between the steps τ and τ', the trace first apply π_1^f (i.e. $T[\tau] \xrightarrow{\text{I-App}(\pi_k^f, f(M_1,\ldots,M_m))}_i T[\tau+1]$) and then deconstruct/reconstruct the term M_1 itself (i.e. $T[\tau+1] \overset{M_1}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$). As previously mentioned, when M_1 is not a data constructor symbol, $T[\tau+1] \overset{M_1}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$ in fact implies $\tau + 1 = \tau'$ meaning that no rule were additionally applied.

Definition 12. *We say that a trace T is* data compliant *when for all $T[\tau] = \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$,*

1. *for all M, if τ is the smallest step such that $M \in \mathcal{A} \setminus \mathsf{data}(T,\tau)$ then $T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$.*
2. *if $T[\tau-1] \rightarrow_i T[\tau]$ by the rule* I-Phase *and $\mathcal{A} = \{M_i\}_{i=1}^m$ then $T[\tau] \overset{M_1}{\Longrightarrow}_{\mathsf{r}} \ldots \overset{M_m}{\Longrightarrow}_{\mathsf{r}} T[\tau']$*
3. *if k is the greatest integer occurring in T then $T[0] \xrightarrow{\text{I-App}(0)}_i \overset{1}{\Longrightarrow}_{\mathsf{r}} \ldots \overset{k}{\Longrightarrow}_{\mathsf{r}} T[k+1]$*

As previously mentioned, we enforce looking at traces where the attacker automatically deconstructs a term with a data constructor and then reconstructs it. The first item of Definition 12 focuses on cases where a term M is known to the attacker for the first time. In such a case, we only look at the trace that applies a complete sequence of deconstruction/reconstruction for the term M (i.e. $T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$). The second item of Definition 12

focuses on phase transition: when the phase of the configuration changes, we require that the trace reconstructs all terms in the attacker knowledge (i.e. $T[\tau] \xRightarrow[\mathsf{r}]{M_1} \ldots \xRightarrow[\mathsf{r}]{M_m} T[\tau']$). Note that it is not necessary to require some deconstruction steps as item 1 of Definition 12 ensures that projections of a term already occur in the attacker knowledge. Finally, the third item of Definition 12 enforces the attacker to "know" the natural numbers that occur in the trace from the beginning. This is possible since both *zero* and *succ* are public functions and *succ* is a data constructor function symbol.

Example 11. Consider the process $P = \mathsf{out}(c, f(a,b)); \mathsf{out}(c,d)$ where f is a data constructor. Let $\rho = \{a \mapsto a[]; b \mapsto b[]; c \mapsto c[]\}$ and let $\mathcal{C} = \rho, P, \{c\}$ be an initial instrumented configuration. In the following, we provide some examples of data compliant traces and non-data compliant traces. For simplicity sake, we only describe the multiset of instrumented of process and the set of terms known by the attacker in instrumented configuration, the other elements remaining unchanged.

- The trace $\mathcal{C} \xrightarrow{\mathrm{msg}(c[],f(a[],b[]))}_i \{\!\{(\mathsf{out}(c,d))\}\!\}, \{c, f(a,b)\}$ is not data compliant
- The following trace T is data compliant:

$$\begin{array}{rl}
\mathcal{C} & \xrightarrow{\mathrm{msg}(c[],f(a[],b[]))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b)\} \\
& \xrightarrow{\text{I-App}(\pi_1^f,f(a,b))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a\} \\
& \xrightarrow{\text{I-App}(\pi_2^f,f(a,b))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a, b\} \\
& \xrightarrow{\text{I-App}(f,a,b)}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a, b\}
\end{array}$$

  In particular, we have $T[1] \xRightarrow[\mathsf{dr}]{f(a,b)} T[4]$ with $T[1] \xRightarrow[\mathsf{d},1]{f(a,b)} T[2] \xRightarrow[\mathsf{d},2]{f(a,b)} T[3] \xRightarrow[\mathsf{r}]{f(a,b)} T[4]$. Note that even though the last step does not change the configuration, it is still a required step for the trace to be data-compliant. The second and third steps deconstruct $f(a,b)$ by applying the projection π_1^f and π_2^f while the fourth step reconstructs the term $f(a,b)$.
- The following trace is not data compliant as item 1 in Definition 12 requires that the deconstruction/reconstruction of terms should not interleaved with other rule applications:

$$\begin{array}{rl}
\mathcal{C} & \xrightarrow{\mathrm{msg}(c[],f(a[],b[]))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b)\} \\
& \xrightarrow{\text{I-App}(\pi_1^f,f(a,b))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a\} \\
& \xrightarrow{\text{I-App}(\pi_2^f,f(a,b))}_i \quad \{\!\{(\mathsf{out}(c,d),\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a, b\} \\
& \xrightarrow{\mathrm{msg}(c[],d[]))}_i \quad \{\!\{(0,\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a, b, d\} \\
& \xrightarrow{\text{I-App}(f,a,b)}_i \quad \{\!\{(0,\emptyset,\emptyset)\}\!\}, \{c, f(a,b), a, b, d\}
\end{array}$$

- Consider the initial instrumented configuration $\mathcal{C}' = \rho, \mathcal{P}, \{c\}$ with $\mathcal{P} = \{\!\{\mathsf{out}(c, f(f(a,2),b)\}\!\}$.

Let $\mathcal{P}_\emptyset = \{\!\{(0, \emptyset, \emptyset)\}\!\}$. The following trace T is data compliant:

$$
\begin{array}{lll}
\mathcal{C}' & \xrightarrow{\text{I-App}(0)}_i & \mathcal{P}, \{c, 0\} \\
& \xrightarrow{\text{I-App}(succ,0)}_i & \mathcal{P}, \{c, 0, 1\} \\
& \xrightarrow{\text{I-App}(succ,1)}_i & \mathcal{P}, \{c, 0, 1, 2\} \\
& \xrightarrow{\text{msg}(c[],f(f(a[],2),b[]))}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b)\} \\
& \xrightarrow{\text{I-App}(\pi_1^f,f(f(a,2),b))}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b), f(a, 2)\} \\
& \xrightarrow{\text{I-App}(\pi_1^f,f(a,2))}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b), f(a, 2), a\} \\
& \xrightarrow{\text{I-App}(f,a,2)}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b), f(a, 2), a\} \\
& \xrightarrow{\text{I-App}(\pi_2^f,f(f(a,2),b))}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b), f(a, 2), a, b\} \\
& \xrightarrow{\text{I-App}(f,f(a,2),b)}_i & \mathcal{P}_\emptyset, \{c, 0, 1, 2, f(f(a, 2), b), f(a, 2), a, b\}
\end{array}
$$

In particular, for the natural numbers $1, 2$ and 3, we have $T[1] \overset{1}{\Rightarrow}_{\mathsf{dr}} T[1]$, $T[2] \overset{2}{\Rightarrow}_{\mathsf{dr}} T[2]$ and $T[3] \overset{3}{\Rightarrow}_{\mathsf{dr}} T[3]$. Furthermore, the steps 4 to 9 is are the deconstruction / reconstruction of $f(f(a, 2), b)$ (i.e. $T[4] \xRightarrow{f(f(a,2),b)}_{\mathsf{dr}} T[9]$) with $T[5] \xRightarrow{f(a,2)}_{\mathsf{dr}} T[7]$ and $T[8] \overset{b}{\Rightarrow}_{\mathsf{dr}} T[8]$. Note that the first steps of the trace consist of generating all integers up to 2 in order to satisfy item 2 of Definition 12. Moreover, notice that data compliance does not require to apply all projections on a data constructor. In particular, we did not apply the step $\xrightarrow{\text{I-App}(\pi_2^f,f(a,2))}_i$ to obtain 2 since it was already in the attacker knowledge, i.e. the step $\xrightarrow{\text{I-App}(\pi_2^f,f(a,2))}_i$ was not required in order to apply $\xrightarrow{\text{I-App}(f,a,2)}_i$. ▶

3.2.2 Internal communications

In the instrumented semantics, the rule I-I/O represents internal communication between two honest processes. When proving a secrecy query, it is well known that such a rule can be ignored when the communication is on a public channel, i.e. we only need to consider traces where all communications on public channels go through the attacker. This is however not the case for general correspondence queries specially when there is an attacker predicate in the conclusion of the query. Indeed, by forcing the communications on public channels to go through the attacker, we potentially increase the knowledge of the attacker thus influencing the satisfaction of attacker facts in the conclusion of a query.

We define a trace to be IO-κ-compliant if internal communications are considered only up to phase $\kappa - 1$.

Definition 13. *Let $\mathcal{C}_0 = \rho_0, P_0, \mathcal{A}_0$ be an initial instrumented configuration. We say that $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$ is* IO-κ-compliant *when T is data compliant and for all steps τ, by denoting $T[\tau] = \kappa', \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$, we have:*

- *if $T[\tau - 1] \xrightarrow{\text{msg}(N,M)}_i T[\tau]$ by the rule* I-I/O *then $\kappa' < \kappa$ or $N \notin \mathcal{A}_0$*
- *if $T[\tau - 1] \xrightarrow{\text{msg}(N,M)}_i T[\tau]$ by the rule* I-Out*, $\kappa' \geq \kappa$ and $N \in \mathcal{A}_0$ then there exists τ' such that $T[\tau] \overset{M}{\Longrightarrow}_{\mathsf{dr}} T[\tau']$ and $T[\tau'] \xrightarrow{\text{msg}(N,M)}_i T[\tau' + 1]$ by the rule* I-Msg*.*

Given an initial instrumented configuration $\mathcal{C}$, we will denote by $\mathsf{trace}^{\kappa}_{IO}(\mathcal{C}, \to_i)$ the set of IO-κ-compliant traces in $\mathsf{trace}(\mathcal{C}, \to_i)$. Given a query ϱ of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi$, we say that ϱ is IO-κ-compliant (resp. fully IO-κ-compliant) when for all attacker facts $\mathrm{att}_{\kappa'}(M)$ occurring in ψ (resp. ψ and $F_1, \ldots, F_n$), we have $\kappa' < \kappa$.

Intuitively, if a query is IO-κ-compliant, that is, it does not prove attacker facts $\mathrm{att}_{\kappa'}(M)$ with $\kappa' \geq \kappa'$, then we need to consider internal communications only up to phase $\kappa - 1$, that is, it is sufficient to consider IO-κ-compliant traces. Note that the second item of the definition is due to T being data compliant. After the transition $T[\tau - 1] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau]$ by the rule I-Out, the attacker knows M but M has not necessarily be de/recomposed. Thus, to ensure that $\mathrm{msg}_{\kappa'}(N, M)$ can be obtained with N and M after their recomposition, we enforce an application of the rule I-Msg.

Example 12. Consider the process $P = \mathsf{new}\ a; \mathsf{out}(c, a) \mid \mathsf{in}^o(c, x)$, $\rho = \{c \mapsto c[]\}$, $\mathcal{A} = \{c\}$ and the initial instrumented configuration $\mathcal{C} = \rho, P, \mathcal{A}$. Consider the query $\varrho = \mathrm{msg}_0(c, x) \Rightarrow \mathrm{att}_0(x)$. Due to a possible internal communication, this query does not hold on P, as witnessed by the following trace T (for readability, we only showed relevant information in the trace).

$$\begin{array}{lll} \mathcal{C} & \to_i & 0, \rho[a' \mapsto a[]], \{\!\{(\mathsf{out}(c, a') \mid \mathsf{in}^o(c, x), \emptyset, \emptyset)\}\!\}, \mathcal{A} \\ & \to_i & 0, \rho[a' \mapsto a[]], \{\!\{(\mathsf{out}(c, a'), \emptyset, \emptyset), (\mathsf{in}^o(c, x), \emptyset, \emptyset)\}\!\}, \mathcal{A} \\ & \xrightarrow{\mathrm{msg}(c[], a[])}_i & 0, \rho[a' \mapsto a[]], \{\!\{(0, \emptyset, \emptyset) \mid (0, [o], [a[]]\}\!\}, \mathcal{A} \end{array}$$

However, if we prevent the application of the internal communication rule I/O on channel c, i.e. we only look at IO-0-compliant traces, then the query would hold. Note that $T \notin \mathsf{trace}^0_{IO}(\mathcal{C}, \to_i)$ but $T \in \mathsf{trace}^1_{IO}(\mathcal{C}, \to_i)$ and ϱ is IO-1-compliant.

Note that the presence of the fact $\mathrm{msg}_0(c, x)$ in the premise of the query ϱ is not mandatory. We can make small modifications on the query and the process that illustrate that only the presence of the fact att_i matters: Consider the process $P_1 = \mathsf{new}\ a; \mathsf{out}(c, a); \mathsf{event}(A(a)) \mid \mathsf{in}^o(c, x)$ and the query $\varrho_1 = (\mathrm{event}(A(x)) \Rightarrow \mathrm{att}_0(x))$. Once again, ϱ_1 does not hold on P_1 but ϱ_1 would hold on P_1 when only looking at IO-0-compliant traces. ▶

3.2.3 Soundness of our restrictions

We previously defined the restricted set of traces we will consider when proving a query, that are the IO-κ-compliant traces. Thus, before showing the soundness, it remains to define the universal satisfaction relation we will consider. $\vdash^{\kappa}_{IO}$ is defined similarly to $\vdash_i$ except that we do not wish to consider the facts that can be introduced by the 2nd bullet of Definition 13.

Definition 14. *Let $\kappa \in \mathbb{N}$. We define the satisfaction relation for IO-κ-compliant traces, denoted $\vdash^{\kappa}_{IO}$, as follows: for all $T \in \mathsf{trace}^{\kappa}_{IO}(\mathcal{C}, \to_i)$, for all steps τ of T, for all facts F, $T, \tau \vdash^{\kappa}_{IO} F$ if and only if $T[\tau] = \kappa', \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and*

- *if $F = \mathrm{msg}_{\kappa''}(N\rho, M\rho)$ and $T[\tau - 1] \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i T[\tau]$ by the rule* I-Out *then $\kappa'' = \kappa'$ and either $\kappa' < \kappa$ or $N \notin \mathcal{A}_0$; and*
- *if $F = \mathrm{att}_{\kappa''}(M\rho)$ then $\kappa'' = \kappa'$ and $M \in \mathsf{data}(T, \tau)$; and*
- *if $F = \mathrm{table}_{\kappa''}(tbl(M_1, \ldots, M_m)\rho)$ then $\kappa'' = \kappa'$ and $tbl(M_1, \ldots, M_m) \in \mathcal{T}$; and*
- *$T, \tau \vdash_i F$ otherwise.*

The following lemma shows the soundness of our restrictions to IO-κ-compliant traces.

Lemma 8. *Let $\mathcal{C}_I = \rho, P, \mathcal{A}$ be an initial instrumented configuration. Let $\kappa \in \mathbb{N}$. Let ϱ be an IO-κ-compliant correspondence query and $\mathcal{R}$ be a set of fully IO-κ-compliant restrictions such that names$(\varrho, \mathcal{R}) \subseteq \mathrm{dom}(\rho)$.*

We have $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}) \models \varrho$ if and only if $(\vdash^\kappa_{IO}, \vdash_i, \mathsf{trace}^\kappa_{IO}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}) \models \varrho$.

3.2.4 Restrictions on bitraces

Similar restrictions can be defined for correspondence queries on bitraces. Since we do not consider injective events for correspondence queries on bitraces, we can keep Definition 10 as it is.

Data compliance To define data compliant bitraces, we need to redefine the set $\mathsf{data}(T, \tau)$ and the relation $\Rightarrow_{\mathsf{dr}}$, $\Rightarrow_{\mathsf{d},k}$ and $\Rightarrow_{\mathsf{r}}$ to consider biterms.

We redefine $\mathsf{data}(T, \tau)$ as a set of pair (M, M') such that $(M, M') \in \mathsf{data}(T, \tau)$ if and only if there exists $\tau' \leq \tau$, $T[\tau'] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and $M'' \in \mathcal{A}$ such that $\mathsf{fst}(M'') = M$, $\mathsf{snd}(M'') = M'$ and if $M = f(M_1, \ldots, M_m)$, $M' = f(M'_1, \ldots, M'_m)$ and $f \in \mathcal{F}_{data}$ then $T[\tau' - 1] \xrightarrow{\text{I-App}(f, \mathsf{diff}[M_1, M'_1], \ldots, \mathsf{diff}[M_m, M'_m])}_{i'} T[\tau']$.

Similarly to the set $\mathsf{data}(T, \tau)$, the de/reconstruction relations will depend on pair of terms, i.e. $\overset{M,M'}{\Longrightarrow}_{\mathsf{dr}}$, $\overset{M,M'}{\Longrightarrow}_{\mathsf{d},k}$ and $\overset{M,M'}{\Longrightarrow}_{\mathsf{r}}$. The main difference comes when M and M' do not have the same data constructor symbol as root. Although this case is not relevant for correspondence queries since such a trace will not be convergent and a correspondence query only considers convergent traces, it is important to preserve traces that are not convergent to prove equivalence. Thus, for all traces T, for all steps τ, we additionally define $T[\tau] \overset{M,M'}{\Longrightarrow}_{\mathsf{dr}} T[\tau]$ (resp. $T[\tau] \overset{M,M'}{\Longrightarrow}_{\mathsf{r}} T[\tau]$, $T[\tau] \overset{M,M'}{\Longrightarrow}_{\mathsf{d},k} T[\tau]$) when $root(M) \neq root(M')$ and either $root(M) \in \mathcal{F}_{data}$ or $root(M') \in \mathcal{F}_{data}$.

The rest of the definition is adapted as expected. For instance, $T[\tau] \overset{M,M'}{\Longrightarrow}_{\mathsf{r}} T[\tau]$ if $(M, M') \in \mathsf{data}(T, \tau)$; and $T[\tau] \overset{f(M_1,\ldots,M_m), f(M'_1,\ldots,M'_m)}{\dashrightarrow}_{\mathsf{r}} T[\tau{+}1]$ if $(M_1, M'_1), \ldots, (M_m, M'_m) \in \mathsf{data}(T, \tau)$ and $T[\tau] \xrightarrow{\text{I-App}(f, \mathsf{diff}[M_1, M'_1], \ldots, \mathsf{diff}[M_m, M'_m])}_{i'} T[\tau + 1]$.

Similarly, Definition 12 is adapted as expected.

IO-κ-compliance No change is required for the notion of IO-κ-compliant, then it only remains to redefine the universal satisfaction relation $\vdash^\kappa_{IO}$ for bifacts. Such a relation is denoted $\vdash^\kappa_{IO'}$ and is defined as follows: $T, \tau \vdash^\kappa_{IO'} F$ if and only if $T[\tau] = \kappa', \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and:

- if $F = \mathrm{msg}'_{\kappa'}(N_1, M_1, N_2, M_2)\rho$ and $T[\tau - 1] \xrightarrow{\mathrm{msg}(N\rho, M\rho)}_i T[\tau]$ by the rule I-Out then $\mathsf{fst}(N) = N_1$, $\mathsf{snd}(N) = N_2$, $\mathsf{fst}(M) = M_1$, $\mathsf{snd}(M) = M_2$ and either $\kappa' < \kappa$ or $N \notin \mathcal{A}_0$; and
- if $F = \mathrm{att}'_{\kappa'}(M_1, M_2)\rho$ then $(M_1, M_2) \in \mathsf{data}(T, \tau)$; and
- if $F = \mathrm{table}'_{\kappa'}(tbl(M_1, \ldots, M_n), tbl(M'_1, \ldots, M'_n))\rho$ then there exists $M \in \mathcal{T}$ such that $\mathsf{fst}(M) = tbl(M_1, \ldots, M_n)$ and $\mathsf{snd}(M) = tbl(M'_1, \ldots, M'_n)$; and

- $T, \tau \vdash_i F$ otherwise.

With these new definitions, we obtain the following two lemmas:

Lemma 9. *Let $\mathcal{C}_I = \rho, P, \mathcal{A}$ be an initial instrumented biconfiguration. Let $\kappa \in \mathbb{N}$. Let $\mathcal{R}$ be a set of fully IO-κ-compliant restrictions such that names$(\mathcal{R}) \subseteq \mathrm{dom}(\rho)$.*

We have $\mathsf{trace}(\mathcal{C}_I, \rightarrow_{i'})_{|\mathcal{R}} \Downarrow\Uparrow_i$ if and only if $\mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|\mathcal{R}} \Downarrow\Uparrow_i$.

Lemma 10. *Let $\mathcal{C}_I = \rho, P, \mathcal{A}$ be an initial instrumented biconfiguration. Let $\kappa \in \mathbb{N}$. Let ϱ be an IO-κ-compliant correspondence query and $\mathcal{R}$ be a set of fully IO-κ-compliant restrictions such that names$(\varrho, \mathcal{R}) \subseteq \mathrm{dom}(\rho)$. We have:*

$$(\vdash_{i'}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_{i'})_{|\mathcal{R}}) \models \varrho \quad \textit{iff} \quad (\vdash^{\kappa}_{IO'}, \vdash_{i'}, \mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|\mathcal{R}}) \models \varrho$$

3.3 Proving correspondence queries by induction

One of the main new features of ProVerif is its ability to prove a correspondence query by induction. This feature applies to all correspondence queries but will be most useful for non-injective queries.

Consider a non-injective IO-κ-compliant correspondence query $\varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \psi)$. Since the query is non-injective, the satisfiability of a correspondence query w.r.t. data compliant steps can be simplified as follows:

> $(\vdash^{\kappa}_{IO}, \vdash_i, \mathcal{S}) \models \varrho$ *if and only if for all $T \in \mathcal{S}$, for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, there exists an annotated query conclusion Ψ w.r.t. (n, m) such that:*
>
> - *$m = \mathsf{max}_{step}(T)$, $\overline{\Psi} = \psi$; and*
> - *for all $\sigma \in \Sigma$, if $T, \tau_i \vdash^{\kappa}_{IO} F_i\sigma$ for $i = 1 \ldots n$ then there exists σ' such that $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma'$.*

Noticeably, this definition ignores the second and third items of Definition 4 since there is no injective event. It also inverses the quantification order between "for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$" and "there exists an annotated query conclusion Ψ ...". In the presence of injective events, it is imperative that the existential quantification over the annotated query conclusion occurs before the universal quantification over the steps $\tilde{\tau}$. This is due to the fact that we need to guarantee in the second item of Definition 4 that the partial functions μ occurring in the annotated query conclusion are injective w.r.t. the injective events.

Formally, to prove that the two definitions are equivalent in the case of non-injective correspondence queries, we only need to focus on showing that the simplified definition implies Definition 4 (the converse being trivial). If we denote by $\Psi_{\tilde{\tau}}$ the existential annotated query conclusion associated to a tuple $\tilde{\tau}$, we can build a new annotated query conclusion $\Psi'_{\tilde{\tau}}$ where each partial function μ in $\Psi'_{\tilde{\tau}}$ is obtained by restricting the domain of the corresponding partial function from $\Psi_{\tilde{\tau}}$ to $\{(\tilde{\tau}, \sigma) \mid \sigma \in \mathcal{S}_\sigma\}$, i.e. μ is defined only on tuples $(\tilde{\tau}, \sigma)$ for any σ. Then, we can build an annotated formula Ψ, that satisfies Definition 4, by composing the corresponding partial functions in all $\Psi'_{\tilde{\tau}}$, for all $\tilde{\tau}$.

We can therefore see the simplified definition as a property of the form: for all $T \in \mathcal{S}$, for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, $\mathcal{P}_\varrho(T, \tilde{\tau})$. ProVerif will thus inductively prove $\mathcal{P}_\varrho(T, \tilde{\tau})$

by considering the strict order relation $<_{ind}$ defined as: $(T, (\tau_1, \ldots, \tau_n)) <_{ind} (T', (\tau'_1, \ldots, \tau'_n))$ if and only if

$$|T| < |T'| \text{ or } (|T| = |T'| \text{ and } \{\!\{\tau_1, \ldots, \tau_n\}\!\} <_m \{\!\{\tau'_1, \ldots, \tau'_n\}\!\})$$

where $<_m$ is the multiset ordering on natural numbers.

Internally, ProVerif will encode the inductive hypothesis as a lemma and will apply it during the saturation and verification procedure only when the strict order is satisfied (contrary to normal lemmas that are always applied).

Group of queries In ProVerif, queries can be declared separately or within groups. In ProVerif 2.00, the two declaration styles affect mainly the saturation procedure: queries declared separately will call the saturation procedure for each query whereas the saturation procedure will be called only once per group of queries. Groups of queries now also influence the inductive proof of queries: if a group of queries, say $\varrho_1, \ldots, \varrho_k$, having $n_1, \ldots, n_k$ facts as premises respectively, has to be proved by induction then the induction hypothesis refers to all queries in the group instead of each individual query. This feature allows to prove queries by mutual induction.

Formally, ProVerif will inductively prove that for all $T \in \mathcal{S}$, for all $\tilde{\tau} \in \mathbb{N}^n$, for all $j \leq k$, for all $(T', \tilde{\tau}') <_{ind} (T, \tilde{\tau})$, $\mathcal{P}_{\varrho_j}(T', \tilde{\tau}')$.

Nested queries As mentioned in Section 2.4, the conclusion of a ProVerif lemma should only contain non-injective events or generic predicates. Thus if the query ϱ contains nested queries, we cannot directly encode the inductive hypothesis as a lemma. We can however *weaken* it so that it can be become a lemma. In particular, we replace any nested condition with a conjunction.

Example 13. The query $\mathsf{event}(A) \Rightarrow \mathsf{event}(B) \rightsquigarrow \mathsf{event}(C)$ implies the query $\mathsf{event}(A) \Rightarrow \mathsf{event}(B) \wedge \mathsf{event}(C)$. ▶

Thus given a query ϱ, if we denote by ϱ_s the simplified query obtained from ϱ by replacing all nested conditions by conjunctions then ProVerif will try to prove that for all $T \in \mathcal{S}$, for all $\tilde{\tau} \in \mathbb{N}^n$, $\mathcal{P}_{\varrho}(T, \tilde{\tau})$ assuming that for all $(T', \tilde{\tau}') <_{ind} (T, \tilde{\tau})$, $\mathcal{P}_{\varrho_s}(T', \tilde{\tau}')$ holds.

Injective queries Similarly to nested queries, we cannot encode directly the injective query into a lemma. Thus, we weaken the inductive hypothesis by replacing any injective events with $\top$, i.e. true.

Example 14. The query $\text{inj}_1\text{-event}(A) \Rightarrow \text{inj}_2\text{-event}(B) \wedge \text{event}(C)$ implies the query $\text{event}(A) \Rightarrow \text{event}(C)$. ▶

Intuitively, the weakened inductive hypothesis only focuses on non injective events as they do not impact the second and third items of Definition 4. Note that in principle, we could replace injective events in the query with events but since we lose all "injectivity" information on the events, they would not help proving the query. In all generality, from a correspondence query ϱ, ProVerif generates a lemma ϱ^{ind} from ϱ by replacing

- all nested conditions by conjunctions
- all facts different from non-injective events in the conclusion of ϱ by $\top$.

The inductive hypothesis that ProVerif will consider is denoted by $\mathcal{IH}_{\varrho^{ind}}$ and is defined as follows.

Definition 15. *Let ϱ be a IO-κ-compliant lemma of the form $\bigwedge_{i=1}^{n} \Rightarrow \psi$. For all traces T, for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, $\mathcal{IH}_{\varrho}(T, \tilde{\tau})$ holds if and only if there exists an annotated query conclusion Ψ w.r.t. (n, m) such that:*

- $m = \mathsf{max}_{step}(T)$, $\overline{\Psi} = \psi$; *and*
- *for all substitutions σ, if $T, \tau_i \vdash^{\kappa}_{IO} F_i\sigma$ for $i = 1 \ldots n$ then there exists σ' such that $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma'$.*

Remark 4. For some queries ϱ, the inductive lemma ϱ^{ind} may not be useful in particular when the conclusion ϱ^{ind} became $\top$ due to injective events. For example, if $\varrho = (\text{inj}_1\text{-event}(A) \Rightarrow \text{inj}_2\text{-event}(B))$ then $\varrho^{ind} = (\mathsf{event}(A) \Rightarrow \top)$ which is always true, hence not useful. Actually, other definitions of ϱ^{ind} could be considered, provided that ϱ entails ϱ^{ind}. ▶

4 Horn clauses generation

As previously mentioned, ProVerif generates from a configuration a set of initial clauses. Moreover, it also generates some Horn clauses to model the behavior of an attacker. We will detail in this section how these clauses are generated. This paper reuses many of the clauses generated by ProVerif described in [BAF08, Bla09, Bla16, CB13].

4.1 Extending the rewrite rules

A conditional rewrite rule is a classic rewrite rule $h(U_1, \ldots, U_n) \to U$ on which a conditional formula ϕ is added, denoted $h(U_1, \ldots, U_n) \to U \mid\mid \phi$. Note that the formula is always of the form $\bigwedge_{i=1}^{n} M_i \geq N_i \wedge \bigwedge_{i'=1}^{n'} \neg nat(M'_i) \wedge \bigwedge_{i''=1}^{n''} nat(M''_i) \wedge \bigwedge_{i'''=1}^{n'''} \forall \tilde{x}_i . M'''_i \neq N'''_i$ where $\tilde{x}$ stands for a sequence of variables. These constraints will be handled by specific rules in the saturation and verification procedure.

Example 15. Assume that $enc/2 \in \mathcal{F}_c$ and consider the destructor $dec/2 \in \mathcal{F}_d$ defined by the list of rewrite rules $\mathsf{def}(dec) = [dec(enc(x, y), y) \to x]$. ProVerif will transform this list of rewrite rules into the following set of conditional rewrite rules:

$$\begin{array}{l} dec(enc(x, y), y) \to x \\ dec(x, y) \to \mathsf{fail} \mid\mid \forall z.x \neq enc(z, y) \\ dec(\mathsf{fail}, u) \to \mathsf{fail} \\ dec(x, \mathsf{fail}) \to \mathsf{fail} \end{array}$$

where x, y are variables and u is a may-fail variable. ▶

Example 16. ProVerif associates the following conditional rewrite rules to the function $geq/2$ on natural numbers:

$$\begin{array}{l} geq(x, y) \to true \mid\mid x \geq y \\ geq(x, y) \to false \mid\mid y \geq x + 1 \\ geq(x, y) \to \mathsf{fail} \mid\mid \neg nat(x) \\ geq(x, y) \to \mathsf{fail} \mid\mid \neg nat(y) \\ geq(\mathsf{fail}, u) \to \mathsf{fail} \\ geq(x, \mathsf{fail}) \to \mathsf{fail} \end{array}$$

▶

Note that ProVerif also associates conditional rewrite rules to constructor function symbols as follows: for all $f/n \in \mathcal{F}_c$,

$$\begin{array}{l} f(x_1, \ldots, x_n) \to f(x_1, \ldots, x_n) \\ f(\mathsf{fail}, u_2, \ldots, u_n) \to \mathsf{fail} \\ f(x_1, \mathsf{fail}, \ldots, u_n) \to \mathsf{fail} \\ \ldots \\ f(x_1, \ldots, x_{n-1}, \mathsf{fail}) \to \mathsf{fail} \end{array}$$

We refer the reader to [CB13] for more details on conditional rewrite rules. In the rest of this paper, we denote $\mathsf{def}(g)$ the set of conditional rewrite rules associated to the function symbol g.

To translate the protocol into clauses, we also need to define evaluation on open terms, as a relation $D \Downarrow' (U, \sigma, \phi)$, where σ collects instantiations of D obtained by unification and ϕ collects the conditional formulae when applying a destructor function symbol. This relation is defined as follows:

$$\begin{array}{l} U \Downarrow' (U, \emptyset, \top) \qquad \text{if } U \text{ is a may-fail term} \\ g(D_1, \ldots, D_n) \Downarrow' (V\sigma_u, \sigma'\sigma_u, \phi'\sigma_u \wedge \phi\sigma_u) \\ \quad \text{if } (D_1, \ldots, D_n) \Downarrow' ((U_1, \ldots, U_n), \sigma', \phi'), \\ \quad g(V_1, \ldots, V_n) \to V \mid\mid \phi \in \mathsf{def}(g) \text{ and} \\ \quad \sigma_u \text{ is the most general unifier of } (U_1, V_1), \ldots, (U_n, V_n) \\ (D_1, \ldots, D_n) \Downarrow' ((U_1\sigma_n, \ldots, U_{n-1}\sigma_n, U_n), \sigma\sigma_n, \phi\sigma_n \wedge \phi_n) \\ \quad \text{if } (D_1, \ldots, D_{n-1}) \Downarrow' ((U_1, \ldots, U_{n-1}), \sigma, \phi) \text{ and } D_n\sigma \Downarrow' (U_n, \sigma_n, \phi_n) \end{array}$$

The most general unifier of may-fail terms U and V is defined as: (i) the standard most general unifier when U and V are in fact both terms; (ii) the substitution $\{U \mapsto V\}$ when U is a may-fail variable; (iii) the identity substitution when $U = V = \mathsf{fail}$. Note that there is no unifier between a term M and fail.

4.2 Clauses generated for correspondence queries

4.2.1 Clauses for the attacker

Below we display the clauses modeling the capabilities of the attacker. They are adapted from [Bla09, CB13]. We consider an initial instrumented configuration $\mathcal{C}_0 = \rho_0, P_0, \mathcal{A}_0$. Moreover, we assume that the maximal phase in P is κ_{max}. Finally, we consider a special name, denoted b_0, that is assumed not to appear in $\mathcal{C}_0$. This name will be used to model all the fresh names that an attacker can create.

For all $\kappa \in \{0, \ldots, \kappa_{max}\}$,

For all $a \in \mathcal{A}_0, \mathrm{att}_\kappa(a[])$	(RInit)
$\mathrm{att}_\kappa(b_0[i])$	(RGen)
$\mathrm{att}_\kappa(\mathsf{fail})$	(RFail)
For all functions h, for all $h(U_1, \ldots, U_n) \to U \mid\mid \phi$ in $\mathsf{def}(h)$ $\mathrm{att}_\kappa(U_1) \wedge \ldots \wedge \mathrm{att}_\kappa(U_n) \wedge \phi \to \mathrm{att}_\kappa(U)$	(Rf)
$\mathrm{msg}_\kappa(x, y) \wedge \mathrm{att}_\kappa(x) \to \mathrm{att}_\kappa(y)$	(Rl)
$\mathrm{att}_\kappa(x) \wedge \mathrm{att}_\kappa(y) \to \mathrm{msg}_\kappa(x, y)$	(Rs)
$\mathrm{att}_\kappa(x) \to \mathrm{att}_{\kappa+1}(x)$	(Rap)
$\mathrm{table}_\kappa(x) \to \mathrm{table}_{\kappa+1}(x)$	(Rtp)

We will denote $\mathbb{C}_\mathcal{A}(\mathcal{C}_0) = \{$(RInit), (RGen), (RFail), (Rf), (Rl), (Rs), (Rap), (Rtp)$\}$.

4.2.2 Clauses for the protocol

As for the attacker clauses, the clauses for the protocol follow closely the ones described in [Bla09, Bla16], except for how to deal with injective events. Once again, we consider an initial instrumented configuration $\mathcal{C}_0 = \rho_0, P_0, \mathcal{A}_0$. In Section 3, we showed that when the query is IO-κ-compliant, we can restrict the search space to IO-compliant traces (see Lemma 8). This property is also reflected in the generation of clauses. Hence we also introduce an integer κ_{io} representing the phase we consider for IO compliance.

The clauses modeling the protocol, denoted $\mathbb{C}_\mathcal{P}(\mathcal{C}_0, \kappa_{io})$, are generated by the translation $[\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho$, displayed in Figure 6 where κ_{io} is the integer used in the translation of output and input case. Note that in Figure 6, $P, \mathcal{O}$ and $\mathcal{I}$ are the same kind of elements as in an instrumented configuration, that are a process, a list of labels and a list of patterns. n represents the current phase. $\mathcal{H}$ is a conjunction of facts and formulas and ρ is a term substitution. Then, we define $\mathbb{C}_\mathcal{P}(\mathcal{C}_0, \kappa_{io})$ as $[\![P_0, \emptyset, \emptyset]\!]0\top\rho_0$.

The transformation of events generates two new type of predicates of arity 2: m-event and s-event respectively called *may-event* and *sure-event*. Intuitively, may-events can only occur in the conclusion of a clause whereas sure-events only occur in the hypotheses. In practice, this separation ensures that events are not "resolved" during the saturation procedure, i.e. once they appear in the hypotheses of a clause, they are preserved through the resolution rule.

For the rest of this paper, we extend the satisfaction relation $\vdash_i$ (and subsequently the relation $\vdash^\kappa_{IO}$) for the predicates m-event and s-event as follows: $T, \tau \vdash_i$ m-event(o, ev) iff $T, \tau \vdash_i$ s-event(o, ev) iff $T, \tau \vdash_i$ event(o, ev).

4.2.3 Soundness

Derivations Since our notion of Horn clauses differs from actual Horn clauses due to the presence of disequalities and inequalities, we need to define the notion of *derivation* of a fact as well as the notion of *subsumption* of clauses. Due to the conditional formulas potentially occurring in $\mathsf{def}(h)$ with $h \in \mathcal{F}_d$, our clauses are of the form $\phi \wedge \bigwedge_{j=1}^{m} F_i \to C$ where $\phi = \bigwedge_{i=1}^{n} M_i \; op_i \; N_i \wedge \bigwedge_{i'=1}^{n'} p_{i'}(M'_i) \wedge \bigwedge_{i''=1}^{n''} \forall \tilde{x}_i . M''_i \neq N''_i$ and $F_1, \ldots, F_n, C$ are facts with $op_i \in \{\geq, =\}$, $p_{i'} \in \{nat, \neg nat\}$. We call ϕ a *constraint formula*. Thus, from now on, when we

$[\![0, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = \emptyset$

$[\![P \mid Q, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho \cup [\![Q, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho$

$[\![!^o P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, (\mathcal{O}, o), (\mathcal{I}, i)]\!]\kappa\mathcal{H}\rho$ where i is a fresh variable session identifier

$[\![\mathsf{new}\ a; P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}(\rho[a \mapsto a[\mathcal{I}]])$

$[\![\mathsf{in}^o(M, x); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, (\mathcal{O}, o), (\mathcal{I}, x')]\!]\kappa(\mathcal{H} \wedge \mathrm{msg}_\kappa(M\rho, x'))(\rho[x \mapsto x'])$
if $(M \notin \mathcal{A}_0$ or $n < n_{IO})$ and x' is a fresh variable

$[\![\mathsf{in}^o(M, x); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, (\mathcal{O}, o), (\mathcal{I}, x')]\!]\kappa(\mathcal{H} \wedge \mathrm{att}_\kappa(x'))(\rho[x \mapsto x'])$
if $M \in \mathcal{A}_0$, $n \geq n_{IO}$ and x' is a fresh variable

$[\![\mathsf{out}(M, N); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{msg}_\kappa(M\rho, N\rho)\}$ if $M \notin \mathcal{A}_0$ or $\kappa < \kappa_{io}$

$[\![\mathsf{out}(M, N); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{att}_\kappa(N\rho)\}$ if $M \in \mathcal{A}_0$ and $\kappa \geq \kappa_{io}$

$[\![\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho =$
$\bigcup\{[\![P, \mathcal{O}, \mathcal{I}\sigma]\!]\kappa(\mathcal{H}\sigma \wedge \phi)(\rho\sigma[x \mapsto M]) \mid D\rho \Downarrow' (M, \sigma, \phi)\}$
$\cup \bigcup\{[\![Q, \mathcal{O}, \mathcal{I}\sigma]\!]\kappa(\mathcal{H}\sigma \wedge \phi)(\rho\sigma) \mid D\rho \Downarrow' (\mathsf{fail}, \sigma, \phi)\}$

$[\![\mathsf{event}^o(ev); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho =$
$[\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\,(\mathcal{H} \wedge \text{s-event}(o[\mathcal{I}], ev\rho))\,\rho \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \text{m-event}(o[\mathcal{I}], ev\rho)\}$

$[\![\mathsf{phase}\ \kappa'; P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa'\mathcal{H}\rho$ if $\kappa' > \kappa$

$[\![\mathsf{phase}\ \kappa'; P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = \emptyset$ if $\kappa' \leq \kappa$

$[\![\mathsf{insert}\ (tbl(M_1, \ldots, M_m)); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho = [\![P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{table}_n(tbl(M_1, \ldots, M_m)\rho)\}$

$[\![\mathsf{get}^o\ T\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho =$
$\{[\![P, (\mathcal{O}, o), (\mathcal{I}\sigma, T\rho'\sigma)]\!]\kappa(\mathcal{H}\sigma \wedge \mathrm{table}_\kappa(T\rho'\sigma) \wedge \phi)(\rho\rho'\sigma) \mid \mathit{equals}(D\rho\rho', \mathit{true}) \Downarrow' (\mathit{true}, \sigma, \phi)\}$
$\cup [\![Q, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho$
where $T = tbl(x_1, \ldots, x_m)$, $\rho' = [x_i \mapsto x_i']_{i=1}^m$ and x_i' fresh variables.

Figure 6: Generation of clauses for correspondence queries

describe a clause in generality, we will write $H \to C$ but we will write $H \wedge \phi \to C$ with H being a conjunction of facts and ϕ being a constraint formula when we want to isolate it from the rest of the hypotheses.

Given two constraint formulas ϕ_1 and ϕ_2, we denote by $\phi_1 \models \phi_2$ when for all substitutions σ, if $\vdash_i \phi_1\sigma$ then $\vdash_i \phi_2\sigma$.

Remark 5. In the implementation, the formulas do not contain equalities as they are directly applied to the clauses. However, for the sake of readability, we make the equalities apparent in this paper as it reduces the number of transformation rules in later sections and makes them more compact. ▶

We define the notion of subsumption as follows.

Definition 16. *Let $H_1 \wedge \phi_1 \to C_1$ and $H_2 \wedge \phi_2 \to C_2$ be two clauses. We say that $H_1 \wedge \phi_1 \to C_1$ subsumes $H_2 \wedge \phi_2 \to C_2$, denoted $(H_1 \wedge \phi_1 \to C_1) \sqsupseteq (H_2 \wedge \phi_2 \to C_2)$, when there exists σ such that (i) either $C_1\sigma = C_2$ or $C_1 = \mathrm{bad}$ (ii) $H_1\sigma \subseteq H_2$ (where H_1 and H_2 are seen as multiset of facts and $\subseteq$ is the multiset inclusion) (iii) $\phi_2 \models \phi_1\sigma$.*

We can now define the notion of derivation and satisfaction of a derivation w.r.t. a trace.

Definition 17. *Let $\mathbb{C}$ be a set of clauses. Let F be a closed fact and a step τ. A derivation $\mathcal{D}$ of F at step τ from $\mathbb{C}$ is a finite tree defined as follows:*

- *its nodes (except the root) are labeled by clauses labels (which may be empty or the pair $\mathcal{O}, \mathcal{I}$) and by clauses $R \in \mathbb{C}$.*
- *its edges are labeled by ground facts and by a step.*
- *if the tree contains a node labeled by R with one incoming edge labeled by F_0 and n outgoing edges labeled by $F_1, \ldots, F_n$ then $R \sqsupseteq F_1 \wedge \ldots \wedge F_n \rightarrow F_0$.*
- *the root has one outgoing edge, labeled by F and τ.*

The previous definition of derivation is an extension of [Bla09, Definition 17]. We only add in our definition the steps of the trace in which the facts should be satisfied. The presence of the trace steps in the derivation is particularly relevant for proving queries by induction and for proving nested queries. Note that in [Bla09], nested queries can already be proved without steps in derivations. However, in this paper, we use the steps, required for proofs by induction, to provide an improved verification algorithm for nested queries.

Definition 18. *Let $\kappa_{io} \in \mathbb{N}$. Let $\mathcal{D}$ be a derivation of F at step τ. Let $\mathcal{S}_p$ be a set of predicates. We say that* a IO-κ_{io}-compliant trace T satisfies a derivation $\mathcal{D}$ w.r.t. $\mathcal{S}_p$, *denoted $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$, when for all nodes η of $\mathcal{D}$, if F_0, τ_0 is the label of the incoming edge of η then*

1. *if η is labeled with* (Rl) *then the outgoing edges of η are labeled $\mathrm{msg}_\kappa(N, M), \tau'$ and $\mathrm{att}_\kappa(N), \tau''$ for some $\kappa, N, M, \tau', \tau''$ such that $\tau' \leq \tau_0$ and if $\mathrm{att}_\kappa \in \mathcal{S}_p$ then $\tau'' < \tau_0$ else $\tau'' \leq \tau_0$.*
2. *if η is not labeled with* (Rl)*, then for all outgoing edges of η labeled F', τ',*
   - *$pred(F') \notin \mathcal{S}_p$ implies $\tau' \leq \tau_0$*
   - *$pred(F') \in \mathcal{S}_p$ implies $\tau' < \tau_0$*
3. *if η is labeled with $(\mathcal{O}, \mathcal{I})$ then there exists $\tau \leq \tau_0$ such that $T[\tau] = \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and $(P, \mathcal{O}, \mathcal{I}) \in \mathcal{P}$.*
4. *if $F_0 = \mathrm{att}_\kappa(f(M_1, \ldots, M_m))$ with $f \in \mathcal{F}_{data}$ then*
   - *either R is the clause $\mathrm{att}_\kappa(x_1) \wedge \ldots \wedge \mathrm{att}_\kappa(x_m) \rightarrow \mathrm{att}_\kappa(f(x_1, \ldots, x_m))$ and if $\mathrm{att}_\kappa \in \mathcal{S}_p$ then $T, \tau_0 \vdash_{IO}^{\kappa_{io}} F_0$;*
   - *or η is not the root and the node η' connected to the incoming edge of η is labeled with the clause $\mathrm{att}_\kappa(f(x_1, \ldots, x_m)) \rightarrow \mathrm{att}_\kappa(x_j)$ for some j and if $\mathrm{att}_\kappa \in \mathcal{S}_p$ then $f(M_1, \ldots, M_m) \in \mathcal{A}$ where $T[\tau_0] = \kappa, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$;*

   else $pred(F_0) \in \mathcal{S}_p$ implies $T, \tau_0 \vdash_{IO}^{\kappa_{io}} F_0$

Intuitively, $\mathcal{S}_p$ represents the set of predicates that we need to prove in the correspondence query. Hence, Item 4 of Definition 18 requires that fact F_0 is satisfied at the step τ_0 in the trace T (e.g. $pred(F_0) \in \mathcal{S}_p$ implies $T, \tau_0 \vdash_{IO}^{\kappa_{io}} F_0$). For all facts that have a predicate not in $\mathcal{S}_p$, we can be looser on the conditions which allow us to apply more transformations on these facts during the saturation procedure. Item 4 is a bit more specific when F_0 is an

attacker fact with a data constructor symbol as root. Typically, either the term has been de/reconstructed in which case we can require that F_0 must satisfied at step τ_0, or the term is being deconstructed at step τ_0 (i.e. the term is used as an argument for its projection functions symbol) in which case we cannot require the satisfaction of F_0 since $\vdash^{\kappa_{io}}_{IO}$ require terms in an attacker predicate to already be reconstructed. Item 2 indicates that facts with a predicate in $\mathcal{S}_p$ must occur strictly before each other (exception in the case of the clause (Rl) as stated in -Item 1). This condition is necessary to apply the inductive lemma. Finally, Item 3 ensures that the derivation corresponds to a valid trace w.r.t. to the session identifier and inputs.

Soundness We can now state the link between traces and derivations. Given a trace $T \in \mathbb{T}(\rightarrow_i)$, we denote by $\mathbb{C}_e(T)$ the set of clauses representing events satisfied in T, i.e. $\mathbb{C}_e(T) = \{\rightarrow \text{s-event}(o, ev) \mid T, \tau \vdash_i \text{s-event}(o, ev)\}$.

Theorem 1. *Let $\mathcal{C}_I = \rho_0, P_0, \mathcal{A}_0$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\kappa_{io} \in \mathbb{N}$.*

For all $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \rightarrow_i)$, for all ground facts F different from a sure-event, for all steps τ, if $T, \tau \vdash^{\kappa_{io}}_{IO} F$ then there exists a derivation $\mathcal{D}$ of F at step τ from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$.

Remark 6. This lemma is very similar to [Bla09, Theorem 1] that states the correctness of the clauses. However, this theorem only states that F is derivable from the set of generated clauses whereas we prove that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$. This property is crucial to ensure that we can apply ProVerif lemmas during the saturation and verification procedure. ▶

4.3 Clauses generated for equivalence queries and correspondence queries on bitraces

Similarly to traces, the generation of clauses for bitraces are parametrized by an initial intermediate biconfiguration $\mathcal{C}_0 = \rho_0, P_0, \mathcal{A}_0$ where we assume that the maximal phase in P_0 is κ_{max}.

4.3.1 Clauses for the attacker

The clauses for the attacker are the same as in [CB13] extended with clauses for phases and tables. The first eight clauses are similar to the clauses for correspondence queries. The last five clauses allow to test if the convergence is not satisfied: intuitively, the predicate $\text{input}'_\kappa(x, y)$ indicates that an input may be available on x and y for the left and right side of the process respectively. On other hand, the predicate $\text{msg}'_\kappa(x, z, y, z')$ indicates that an output may be available on x and y for the left and right side respectively. Therefore, if the hypotheses of clause (RIBad1') are satisfied then it implies that the first condition of definition 6 may be broken.

Similarly, the hypotheses of clauses (RBad1') and (RBad2') can only be derivable if the evaluation of an expression failed on one side of the process but not on the other. In such case, the second condition of Definition 6 would not be satisfied.

$$\begin{array}{l}\text{For all } \kappa \in \{0, \ldots, \kappa_{max}\},\\ \quad \text{For each } a \in \mathcal{A}_0, \text{att}'_\kappa(a[], a[])\end{array} \qquad \text{(RInit')}$$

$$\mathrm{att}'_\kappa(b_0[i], b_0[i]) \qquad \text{(RGen')}$$
$$\mathrm{att}'_\kappa(\mathsf{fail}, \mathsf{fail}) \qquad \text{(RFail')}$$
$$\begin{array}{l}\text{For each function } h, \text{ for all } h(U_1, \ldots, U_m) \to U \mid\mid \phi \text{ in } \mathsf{def}(h),\\ \quad \text{for all } h(U'_1, \ldots, U'_m) \to U' \mid\mid \phi' \text{ in } \mathsf{def}(h), \qquad \text{(Rf')}\\ \quad \mathrm{att}'_\kappa(U_1, U'_1) \wedge \ldots \wedge \mathrm{att}'_\kappa(U_m, U'_m) \wedge \phi \wedge \phi' \to \mathrm{att}'_\kappa(U, U')\end{array}$$
$$\mathrm{msg}'_\kappa(x, y, x', y') \wedge \mathrm{att}'_\kappa(x, x') \to \mathrm{att}'_\kappa(y, y') \qquad \text{(Rl')}$$
$$\mathrm{att}'_\kappa(x, x') \wedge \mathrm{att}'_\kappa(y, y') \to \mathrm{msg}'_\kappa(x, y, x', y') \qquad \text{(Rs')}$$
$$\mathrm{att}'_\kappa(x, y) \to \mathrm{att}'_{\kappa+1}(x, y) \qquad \text{(Rap')}$$
$$\mathrm{table}'_\kappa(x, y) \to \mathrm{table}'_{\kappa+1}(x, y) \qquad \text{(Rtp')}$$
$$\mathrm{att}'_\kappa(x, x') \to \mathrm{input}'_\kappa(x, x') \qquad \text{(RIn')}$$
$$\mathrm{input}'_\kappa(x, y) \wedge \mathrm{msg}'_\kappa(x, z, y', z') \wedge y \neq y' \to \mathrm{bad} \qquad \text{(RIBad1')}$$
$$\mathrm{input}'_\kappa(y, x) \wedge \mathrm{msg}'_\kappa(y', z, x, z') \wedge y \neq y' \to \mathrm{bad} \qquad \text{(RIBad2')}$$
$$\mathrm{att}'_\kappa(x, \mathsf{fail}) \to \mathrm{bad} \qquad \text{(RBad1')}$$
$$\mathrm{att}'_\kappa(\mathsf{fail}, x) \to \mathrm{bad} \qquad \text{(RBad2')}$$

We denote $\mathbb{C}'_{\mathcal{A}}(\mathcal{C}_0) = \{$ (RInit'), (RGen'), (RFail'), (Rf'), (Rs'), (Rl'), (RIn'), (RIBad1'), (RIBad2'), (RBad1'),(RBad2'), (Rap'), (Rtp') $\}$.

4.3.2 Clauses for the protocol

Similarly to the case of correspondence properties, we consider traces with IO compliance for some phase κ_{io}. Moreover, the translation also introduces two new predicates of arity 2: s-event$'$ and m-event$'$ that are the sure-event and may-event for bitraces. The clauses modeling the protocol, denoted $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_0, \kappa_{io})$, are generated by the translation $[\![|P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho$ where ρ is a mapping from variables to terms containing potentially diff, $\mathcal{H}$ is a conjunction of facts and formulae. Then, we have $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_0, \kappa_{io}) = [\![|P_0, \emptyset, \emptyset|]\!]0\top\rho_0$.

$[\![|0, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho = \emptyset$

$[\![|!^o P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P, (\mathcal{O}, o), (\mathcal{I}, i)|]\!]\kappa\mathcal{H}\rho$ where i is a fresh variable

$[\![|P \mid Q, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho \cup [\![|Q, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho$

$[\![|\mathsf{new}\ a; P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}(\rho[a \mapsto a[\mathcal{I}]])$

$[\![|\mathsf{in}^o(N, x); P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho =$

$\quad [\![|P, (\mathcal{O}, o), (\mathcal{I}, \mathsf{diff}[x_1, x_2])|]\!]\kappa(\mathcal{H} \wedge \mathrm{msg}'_n(\mathsf{fst}(N\rho), x_1, \mathsf{snd}(N\rho), x_2))(\rho[x \mapsto \mathsf{diff}[x_1, x_2]])$

$\quad \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{input}'_\kappa(\mathsf{fst}(N\rho), \mathsf{snd}(N\rho))\}$ if ($N \notin \mathcal{A}_0$ or $\kappa < \kappa_{io}$) and x_1, x_2 fresh

$[\![|\mathsf{in}^o(N, x); P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho =$

$\quad [\![|P, (\mathcal{O}, o), (\mathcal{I}, \mathsf{diff}[x_1, x_2])|]\!]\kappa(\mathcal{H} \wedge \mathrm{att}'_\kappa(x_1, x_2))(\rho[x \mapsto \mathsf{diff}[x_1, x_2]])$

$\quad$ if $N \in \mathcal{A}_0$, $\kappa \geq \kappa_{io}$ and x_1, x_2 fresh

$[\![|\mathsf{out}(N, M); P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P, \mathcal{O}, \mathcal{I}|]\!]\kappa\mathcal{H}\rho$

$\quad \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{msg}'_\kappa(\mathsf{fst}(N\rho), \mathsf{fst}(M\rho), \mathsf{snd}(N\rho), \mathsf{snd}(M\rho))\}$ if $N \notin \mathcal{A}_0$ or $\kappa < \kappa_{io}$

$[\![|\mathsf{out}(N,M);P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho \cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{att}'_\kappa(\mathsf{fst}(M\rho),\mathsf{snd}(M\rho))\}$
if $N \in \mathcal{A}_0$ and $\kappa \geq \kappa_{io}$

$[\![|\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho =$

$\bigcup\{[\![|P,\mathcal{O},\mathcal{I}\sigma|]\!]\kappa(\mathcal{H}\sigma\wedge\phi)(\rho\sigma[x\mapsto\mathsf{diff}[pt_1,pt_2]]) \mid (\mathsf{fst}(D\rho),\mathsf{snd}(D\rho)) \Downarrow' ((pt_1,pt_2),\sigma,\phi)\}$

$\cup \bigcup\{[\![|Q,\mathcal{O},\mathcal{I}\sigma|]\!]\kappa(\mathcal{H}\sigma\wedge\phi)(\rho\sigma) \mid (\mathsf{fst}(D\rho),\mathsf{snd}(D\rho)) \Downarrow' ((\mathsf{fail},\mathsf{fail}),\sigma,\phi)\}$

$\cup \{\mathcal{H}\sigma\wedge\phi \xrightarrow{\mathcal{O},\mathcal{I}\sigma} \mathrm{bad} \mid ((\mathsf{fst}(D\rho),\mathsf{snd}(D\rho)) \Downarrow' ((\mathsf{fail},pt),\sigma,\phi)\}$

$\cup \{\mathcal{H}\sigma\wedge\phi \xrightarrow{\mathcal{O},\mathcal{I}\sigma} \mathrm{bad} \mid ((\mathsf{fst}(D\rho),\mathsf{snd}(D\rho)) \Downarrow' ((pt,\mathsf{fail}),\sigma,\phi)\}$

$[\![|\mathsf{event}^o(ev);P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P,\mathcal{O},\mathcal{I}|]\!]\kappa\left(\mathcal{H}\wedge\text{s-event}'(\mathsf{fst}(ev\rho),\mathsf{snd}(ev\rho))\right)\rho$

$\cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \text{m-event}'(\mathsf{fst}(ev\rho),\mathsf{snd}(ev\rho))\}$

$[\![|\mathsf{phase}\ \kappa';P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P,\mathcal{O},\mathcal{I}|]\!]\kappa'\mathcal{H}\rho \qquad \text{if } \kappa' > \kappa$

$[\![|\mathsf{phase}\ \kappa';P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho = \emptyset \qquad \text{if } \kappa' \leq \kappa$

$[\![|\mathsf{insert}\ tbl(M_1,\ldots,M_k);P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho = [\![|P,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho$

$\cup \{\mathcal{H} \xrightarrow{\mathcal{O},\mathcal{I}} \mathrm{table}'_\kappa(tbl(\mathsf{fst}(M_1\rho),\ldots,\mathsf{fst}(M_k\rho),tbl(\mathsf{snd}(M_1\rho),\ldots,\mathsf{snd}(M_k\rho))\}$

$[\![|\mathsf{get}^o\ T\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho =$

$\bigcup\{[\![|P,(\mathcal{O},o),(\mathcal{I}\sigma,T\rho'\sigma)|]\!]\kappa(\mathcal{H}\sigma\wedge\phi)(\rho\rho'\sigma) \mid (\mathsf{fst}(D\rho\rho'),\mathsf{snd}(D\rho\rho')) \Downarrow' ((true,true),\sigma,\phi)\}$

$\cup \{\mathcal{H}\sigma\wedge\phi\wedge u\neq true \xrightarrow{\mathcal{O},\mathcal{I}\sigma} \mathrm{bad} \mid ((\mathsf{fst}(D\rho\rho'),\mathsf{snd}(D\rho\rho')) \Downarrow' ((true,u),\sigma,\phi)\}$

$\cup \{\mathcal{H}\sigma\wedge\phi\wedge u\neq true \xrightarrow{\mathcal{O},\mathcal{I}\sigma} \mathrm{bad} \mid ((\mathsf{fst}(D\rho\rho'),\mathsf{snd}(D\rho\rho')) \Downarrow' ((u,true),\sigma,\phi)\}$

$\cup [\![|Q,\mathcal{O},\mathcal{I}|]\!]\kappa\mathcal{H}\rho$

where $T = tbl(x_1,\ldots,x_m)$, $\rho' = [x_i \mapsto \mathsf{diff}[x'_i,x''_i]]_{i=1}^m$ and x'_i, x''_i are fresh.

Remark 7. With respect to [CB13], we formally define the translation of phases and tables. Similarly to correspondence queries, we cannot express in Horn clauses that an element is not in the table. Thus, the conditions on the table are ignored when executing the else branch. ▶

4.3.3 Soundness

For biprocesses, we establish two soundness results: one with respect to the convergence of bitraces, another one with respect to correspondence queries on bitraces. Once again, we adapt as expected the satisfaction relation $\vdash$ for derivation with bifacts, that we denote $\vdash'$. In particular, items 2 and 1 of Definition 18 depend on the clause (Rl') instead of (Rl).

Soundness for convergence of bitraces. Similarly to our definition of $\mathbb{C}_e(T)$ on traces, we define $\mathbb{C}'_e(T)$ on bitraces. For instance, if $T \in \mathbb{T}(\rightarrow_{i'})$ then $\mathbb{C}'_e(T) = \{\text{s-event}'(ev,ev') \mid T,\tau \vdash_{i'} \text{s-event}'(ev,ev')\}$.

The following lemma shows the soundness of the non-derivability of bad.

Lemma 11. *Let $\mathcal{C}_I = \rho_0,P_0,\mathcal{A}_0$ be an initial instrumented biconfiguration. Let $\mathcal{S}_p$ be a set of predicates such that* bad $\notin \mathcal{S}_p$. *Let $\kappa_{io} \in \mathbb{N}$.*

For all $T \in \mathsf{trace}_{IO}^{\kappa_{io}}(\mathcal{C}_I, \rightarrow_{i'})$*, if* τ *is the maximal step in* T *and* $\neg T{\downarrow\uparrow_i}$ *then there exists a derivation* $\mathcal{D}$ *of* bad *from* $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{e'}^{<\tau}(T)$ *such that* $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}$*.*

Remark 8. This lemma is very similar to [CB13, Theorem 1] that states the correctness of the clauses. Similarly to the case of correspondence queries, this theorem only states that *bad* is derivable from the set of generated clauses whereas we prove that $T, \mathcal{S}_p, 0 \vdash' \mathcal{D}$. This is once again required to apply ProVerif lemmas. ▶

For correspondence queries on bitraces, we can restrict the set of clauses we consider. All clause of the form $H \rightarrow$ bad in $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$ and $\mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$ aim to check whether a bitrace diverges. Thus we can discard the clauses $H \rightarrow$ bad. Let us denote $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io})$ be the set of clauses in $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$ that does not have bad as conclusion. We obtain the following soundness result.

Lemma 12. *Let* $\mathcal{C}_I = \rho_0, P_0, \mathcal{A}_0$ *be an initial instrumented biconfiguration. Let* $\mathcal{S}_p$ *be a set of predicates. Let* $\kappa_{io} \in \mathbb{N}$*.*

For all $T \in \mathsf{trace}_{IO}^{\kappa_{io}}(\mathcal{C}_I, \rightarrow_{i'})$*, for all ground bifacts* F *different from a sure-event, for all steps* τ*, if* $T, \tau \vdash_{IO'}^{\kappa_{io}} F$ *then there exists a derivation* $\mathcal{D}$ *of* F *at step* τ *from* $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_e(T)$ *such that* $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}$*.*

4.4 Precise actions

We showed in Lemma 2 that all tuples $(\mathcal{O}, \mathcal{I})$ found in the instrumented semantics to be compatible with one another. However, in the saturation procedure, ProVerif may build derivations that do not satisfy this property which may lead to false attacks.

Example 17. Consider $P_0 = \mathsf{new}\ k; \mathsf{out}(c, senc(senc(s,k),k)); \mathsf{in}^o(c,x); \mathsf{out}(c, sdec(x,k))$, $\rho_0 = [c \mapsto c[]; s \mapsto s[]]$ and the initial instrumented configuration $\mathcal{C}_I = \rho_0, P_0, \{c\}$. The function symbols *senc* and *sdec* respectively are the symmetric encryption and decryption. $\mathcal{C}_I$ satisfies the secrecy of s. Intuitively, the process allows the intruder to decrypt only one cypher encrypted with the key k but not two. Thus, the intruder cannot decrypt $senc(senc(s,k),k)$. However, from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, 0)$, we can build the following derivation:

$$
\begin{array}{r}
\mathrm{att}_0(s[]) \\
\downarrow \\
(o), (senc(s[], k[])) \quad \boxed{\mathrm{att}_0(senc(x, k[]) \xrightarrow{(o),(senc(x,k[]))} \mathrm{att}_0(x)} \\
\mathrm{att}_0(senc(s[], k[])) \\
\downarrow \\
(o), (senc(senc(s[], k[]), k[])) \quad \boxed{\mathrm{att}_0(senc(x, k[]) \xrightarrow{(o),(senc(x,k[]))} \mathrm{att}_0(x)} \\
\mathrm{att}_0(senc(senc(s[], k[]), k[])) \\
\downarrow \\
\emptyset, \emptyset \quad \boxed{\xrightarrow{(\emptyset,\emptyset)} \mathrm{att}_0(senc(senc(s[], k[]), k[]))}
\end{array}
$$

This derivation will be considered by ProVerif meaning that ProVerif will indicate that it is not able to prove the secrecy of s. However, one can notice that the two tuples $((o), (senc(s[], k[])))$ and $((o), (senc(senc(s[], k[]), k[])))$ are not compatible hence this derivation cannot correspond to a real trace of the instrumented semantics. ▶

To prevent ProVerif from failing on such cases, the set of clauses $\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$ is augmented with an event *precise* when translating inputs and table lookups as follows:

$[\![\mathsf{in}^o(M,x); P, \mathcal{O}, \mathcal{I}]\!]\kappa\mathcal{H}\rho =$
$\quad [\![P, (\mathcal{O}, o), (\mathcal{I}, x')]\!]\kappa(\mathcal{H} \wedge \text{s-event}(o[\overline{\mathcal{I}}], \mathit{precise}(x')) \wedge \text{msg}_\kappa(M\rho, x'))(\rho[x \mapsto x'])$
where x' is a fresh variable and $\overline{\mathcal{I}}$ is obtained from $\mathcal{I}$ by keeping only the session identifiers

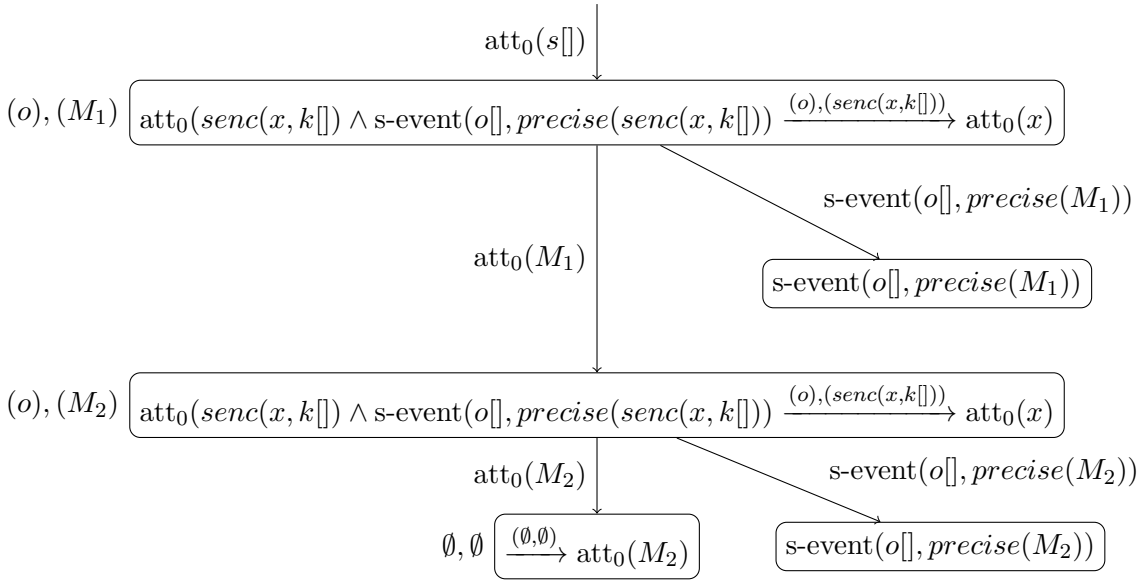
$[\![\mathsf{get}^o\ T\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \kappa, \mathcal{O}, \mathcal{H}, \mathcal{I}]\!]\rho =$
$\quad \{[\![P, (\mathcal{O}, o), (\mathcal{I}\sigma, T\rho'\sigma)]\!]\kappa(\mathcal{H}\sigma \wedge \text{s-event}(o[\overline{\mathcal{I}}], \mathit{precise}(T\rho'\sigma)) \wedge \text{table}_\kappa(T\rho'\sigma) \wedge \phi)(\rho\rho'\sigma) \mid$
$\qquad \mathit{equals}(D\rho\rho', \mathit{true}) \Downarrow' (\mathit{true}, \sigma, \phi)\}$
$\quad \cup [\![Q, \kappa, \mathcal{O}, \mathcal{H}, \mathcal{I}]\!]\rho$
$\quad$ where $T = \mathit{tbl}(x_1, \ldots, x_m)$, $\rho' = [x_i \mapsto x_i']_{i=1}^m$, x_i' fresh variables and $\overline{\mathcal{I}}$ is obtained from $\mathcal{I}$ by keeping only the session identifiers

To ensure that ProVerif only considers derivations that satisfy compatibility, we rely on the newly introduced events *precise* and the following axiom:

$$\text{event}(o, \mathit{precise}(x)) \wedge \text{event}(o, \mathit{precise}(x')) \Rightarrow x = x'$$

This axiom indicates that whenever the derivation contains two events *precise* with the same occurrence, they should also have the same input messages.

Example 18. Continuing Example 17, we obtain the following derivation with the augmented set of clauses $\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, 0)$:



with $M_1 = \mathit{senc}(s[], k[])$ and $M_2 = \mathit{senc}(\mathit{senc}(s[], k[]), k[])$. On this derivation, the application of the axiom will allow ProVerif to find a contradiction since M_1 and M_2 cannot be equal. Hence the derivation will be discarded. ▶

Remark 9. In ProVerif 2.00, the compatibility property is only taken into account during an attack reconstruction but not during the saturation procedure. ▶

Remark 10. The *precise* axiom is the only built-in axiom in ProVerif as we provide a proof in this paper of its correctness. Note that ProVerif does not prove any of the user-defined axioms. ▶

5 Saturation procedure

In this section, we describe the saturation procedure we use to prove the existence of a derivation for bad or facts in correspondence queries. We will first explain the resolution rule and selection function that are at the heart of the procedure. Second, we will present the classic simplification rules that are already used by ProVerif. Third, we will present our new simplifications rules based on lemmas. Finally, we will show how the saturation procedure handles inductive queries. With this final set of simplifications rules, one can note that the saturation procedure will from now depend on the query.

Note that in the implementation, ProVerif uses two different saturation procedures, one for equivalence queries and another one for correspondence queries. While both of them are based on the same principle, i.e. generating new rules by resolution, they apply different sets of simplification rules. Therefore, when describing the latter, we will clarify whether these simplifications are for correspondence or equivalence queries.

Our saturation procedure removes the labels $(\mathcal{O}, \mathcal{I})$ on clauses in $\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$ and $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$. Intuitively, since we have recorded the information they provide within event *precise* (see Section 4.4), we don't need them in the saturation procedure.

5.1 Resolution rule and selection function

The saturation procedure will generate new clauses by combining two existing clauses. Assume that $\mathbb{C}$ is a set of clauses, the resolution rule proceeds as follows:

$$\frac{H \to C \in \mathbb{C} \qquad F \wedge H' \to C' \in \mathbb{C} \qquad \sigma = mgu(F, C)}{H\sigma \wedge H'\sigma \to C'\sigma} \text{ (Res)}$$

The resolution rule allows to shorten the size of the derivations, e.g. a derivation for an instance $C'\sigma$ could directly use the newly generated rule instead of the two rules $H \to C$ and $F \wedge H' \to C'$ separately. To avoid trivial termination issues, the application of (Res) is restricted by a selection function. In particular, the resolution procedure is parameterized by a set of facts that cannot be selected.

Definition 19. *We say that a fact F is* unselectable *when F is of the form* $\mathrm{att}_i(u)$, $\text{s-event}(ev)$ *or* $\text{s-event}'(ev)$ *for all (may-fail) variables u, for all $i \in \mathbb{N}$ and for all events ev. We denote by $\mathbb{F}_{usel}$ the set of unselectable facts.*

We say that sel *is a selection function for $\mathbb{F}_{usel}$ when* sel *is a function from clauses to sets of facts such that:*

- $\mathrm{sel}(H \to C) \subseteq H \setminus \mathbb{F}_{usel}$
- $\mathrm{att}'_\kappa(x, x') \in \mathrm{sel}(H \wedge \phi \to C)$ *implies that x or x' is a variable of ϕ and for all $F \in H \setminus \mathbb{F}_{usel}$, $F = \mathrm{att}'_{\kappa'}(y, y')$ for some y, y', κ'.*
- *If $C \in \mathbb{F}_{usel}$ and $\mathrm{sel}(H \to C) = \emptyset$ then for all facts $F \in H$, $F \in \mathbb{F}_{usel}$.*

For the rest of this paper, we will fix a set of unselectable facts $\mathbb{F}_{usel}$ and a selection function sel. The resolution rule is applied only when $\mathrm{sel}(H \to C) = \emptyset$ and $F \in \mathrm{sel}(F \wedge H' \to C')$.

Remark 11. The main goal of the saturation procedure is to apply the resolution rule until reaching a fixpoint. Thus, the more restrictive the selection function sel is, the more likely the procedure will terminate. However, it also reduces the precision of ProVerif since it will stop resolving fact in more cases. This is one of the reason the default selection function focuses only on may-events and attacker facts containing only variables: May-events by definition cannot be solved; an attacker fact with just a variable indicates that the attacker can generate any term to instantiate the variable. In particular the attacker can replace the variable by a fresh name, which would not have required any resolution. ▶

5.2 Classic simplification rules

After generating a new clause by resolution, ProVerif applies multiple simplification rules. Some of them are standard and hold for both equivalence and correspondence queries.

$$\frac{\mathbb{C} \cup \{F \wedge H \to F\}}{\mathbb{C}} \text{ (Taut)}$$

$$\frac{\mathbb{C} \cup \{H' \wedge H \wedge \phi \to C\} \qquad H'\sigma \subseteq H \qquad \phi \models \phi\sigma \qquad \mathrm{dom}(\sigma) \cap fv(H, C) = \emptyset}{\mathbb{C} \cup \{H \wedge \phi\sigma \to C\}} \text{ (Red)}$$

$$\frac{\mathbb{C} \cup \{\mathrm{att}_\kappa(x) \wedge H \wedge \phi \to C\} \qquad x \text{ does not appear in } H, C}{\mathbb{C} \cup \{H \wedge \phi \to C\}} \text{ (Att)}$$

$$\frac{\mathbb{C} \cup \{\mathrm{att}'_\kappa(x, y) \wedge H \wedge \phi \to C\} \qquad x \text{ and } y \text{ do not appear in } H, C}{\mathbb{C} \cup \{H \wedge \phi \to C\}} \text{ (Att')}$$

The rule (Taut) removes a clause that has a fact as both hypothesis and conclusion. The rule (Red) removes redundant facts from the hypotheses of a clause. The rule (Att) and the equivalent rule for bitraces (Att') removes a fact $\mathrm{att}_\kappa(x)$ where the variable x does not occur anywhere else in the clause other than in the constraint formula. Intuitively, we can always obtain an equivalent derivation where we replace x by any fresh name from the attacker or by a natural number that satisfies the constraint formula. As previously mentioned, ProVerif always applies the most general unifier of equalities in the formula when it exists or discards the clause otherwise. ProVerif also considers transformations for simplifying the disequalities. These transformations only modify the formulas in the clause and preserve their solutions. In particular, it discards the clause when the disequalities are not satisfiable. We will regroup these transformations under one rule that we call (R_ϕ). We refer the reader to [BAF08, Bla16] for more details.

Vanilla ProVerif also considers specific transformation rules to deal with data constructor function symbols. By definition of the clause (Rf), we know that for all $g/m \in \mathcal{F}_{data}$, for all phases n, the following clauses are in $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$.

$$\mathrm{att}_\kappa(x_1) \wedge \ldots \wedge \mathrm{att}_\kappa(x_m) \to \mathrm{att}_\kappa(g(x_1, \ldots, x_n)) \qquad (\mathrm{Rf}_g)$$
$$\mathrm{att}_\kappa(g(x_1, \ldots, x_m)) \to \mathrm{att}_\kappa(x_i) \qquad \text{for all } i \in \{1, \ldots, m\} \qquad (\mathrm{Rf}_{\pi_i^g})$$

Hence on a clause $R = H \wedge \mathrm{att}_\kappa(g(M_1, \ldots, M_m)) \to C$ where $g \in \mathcal{F}_{data}$, we can apply the resolution between R and Rf_g. This would lead to the rule $H \wedge \mathrm{att}_\kappa(M_1) \wedge \ldots \wedge \mathrm{att}_\kappa(M_m) \to C$. Similarly, on a clause $R = H \to \mathrm{att}_\kappa(g(M_1, \ldots, M_m))$ we can apply the resolution between R and $\mathrm{Rf}_{\pi_i^g}$ leading to the rules $H \to \mathrm{att}_\kappa(M_i)$ for all $i \in \{1, \ldots, m\}$. To speed up the saturation procedure, we always apply these resolutions, which lead to the following transformation rules.

$$\frac{\mathbb{C} \cup \{R = (\mathrm{att}_\kappa(g(M_1, \ldots, M_m)) \wedge H \to C)\} \qquad g \in \mathcal{F}_{data} \qquad R \neq (\mathrm{Rf}_{\pi_i^g}) \text{ for all } i}{\mathbb{C} \cup \{\mathrm{att}_\kappa(M_1) \wedge \ldots \wedge \mathrm{att}_\kappa(M_m) \wedge H \to C\}} \text{(DataHyp)}$$

$$\frac{\mathbb{C} \cup \{R = (H \to \mathrm{att}_\kappa(g(M_1, \ldots, M_m)))\} \qquad g \in \mathcal{F}_{data} \qquad R \neq (\mathrm{Rf}_g)}{\mathbb{C} \cup \{H \to \mathrm{att}_\kappa(M_i)\}_{i=1}^m} \text{(DataCl)}$$

As usual, we have a version of these two transformation rules for bitraces that we denote DataCl' and DataHyp' where bifacts $\mathrm{att}'_\kappa(g(M_1, \ldots, M_m), g(N_1, \ldots, N_m))$ are split into $\mathrm{att}'_\kappa(M_1, N_1)$, $\ldots$, $\mathrm{att}'_\kappa(M_m, N_m)$.

5.3 General redundancy

The rule Red focuses on redundant facts within a clause. More generally, we also consider a transformation that checks whether a clause itself is redundant with respect to the rest of the clauses with no hypothesis selectable. Intuitively, we will remove a clause R with $\mathrm{sel}(R) = \emptyset$ if for all derivations using R, we can build another derivation for the same fact that does not use the clause R. We formally define this notion by introducing *partial derivations*.

Definition 20. *Let $\mathbb{C}$ be a set of clauses. Let F be a fact (not necessarily closed). A* partial derivation *$\mathcal{D}$ of F from $\mathbb{C}$ is a finite tree defined as follows:*

- *its nodes can either be unlabeled or be labeled by clauses $R \in \mathbb{C}$. Furthermore, the root is never labeled and all internal nodes (i.e. not the root nor leaves) are labeled.*
- *incoming edges of unlabeled leaves can be labeled by facts or formulae. All other edges are only labeled by facts.*
- *if the tree contains a node labeled by R with one incoming edge labeled by F_0 and n outgoing edges labeled by $H_1, \ldots, H_n$ (possibly facts or formulae) then $R \sqsupseteq H_1 \wedge \ldots \wedge H_n \to F_0$.*
- *the root has one outgoing edge, labeled by F.*
- *a node cannot be labeled by a clause $\mathrm{att}_\kappa(x) \wedge H \to \mathrm{att}_{\kappa'}(x)$ where $H \neq \emptyset$ or $\kappa' \neq \kappa + 1$.*

We denote by $\mathbb{F}_{us}(\mathcal{D})$ the set of all facts and formulae labeling the incoming edges of unlabeled leaves. We also denote by $\mathbb{F}_s(\mathcal{D})$ the set of all facts labeling the incoming edges of labeled nodes.

Intuitively, the leaves without label are facts that haven't been solved by resolution yet. Moreover, the last item of the definition indicates that the phase can only be changed by application of the clause Rap.

The general redundancy transformation rule is thus defined as follows:

$$\frac{\begin{array}{c}\mathbb{C} \cup \{H \to F\} \qquad \mathbb{C}' = \{R' \in \mathbb{C} \mid \mathrm{sel}(R') = \emptyset\} \qquad \mathrm{sel}(H \to F) = \emptyset \\ \exists\, \mathcal{D} \text{ partial derivation of } F \text{ from } \mathbb{C}' \text{ s.t.} \\ \mathbb{F}_{us}(\mathcal{D}) \to F \sqsupseteq H \to F \text{ and } \mathit{pred}(\mathbb{F}_s(\mathcal{D})) \cap \mathcal{S}_p = \emptyset\end{array}}{\mathbb{C}} \text{(GRed}(\mathcal{S}_p)\text{)}$$

Note that the rule is parametrized by the set of predicates $\mathcal{S}_p$. This is the same set we use in Theorem 1 to prove the soundness of the initial set of generated clauses w.r.t. the instrumented semantics. As previously mentioned, the set $\mathcal{S}_p$ represents the set of predicates that we need to prove in the correspondence query. For the facts with these predicates, it is crucial that we preserve the order in which they are satisfied in the trace (as stated in Definition 18). However, in the rule $\text{GRed}(\mathcal{S}_p)$, when we replace a clause $H \to F$ by the derivation $\mathcal{D}$, there is no guarantee that the facts in $\mathbb{F}_s(\mathcal{D})$ are satisfied by the trace. Therefore, we only apply the transformation if $\mathit{pred}(\mathbb{F}_s(\mathcal{D})) \cap \mathcal{S}_p = \emptyset$.

Specific rules for equivalence properties As previously mentioned, we assume that the destructor *equals* is defined by the following set $\mathsf{def}(\mathit{equals})$:

$$\left\{ \begin{array}{cc} \mathit{equals}(x,x) \to x; & \mathit{equals}(\mathsf{fail},u) \to \mathsf{fail} \\ \mathit{equals}(u,\mathsf{fail}) \to \mathsf{fail}; & \mathit{equals}(x,y) \to \mathsf{fail} \mid\mid x \neq y \end{array} \right\}$$

Therefore, in the clauses for the attacker, two of the instances of the clauses (Rf') will be:

$$\begin{array}{c} \mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_\kappa(x,y') \wedge x' \neq y' \to \mathrm{att}'_\kappa(x,\mathsf{fail}) \\ \mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_\kappa(y,x') \wedge x \neq y \to \mathrm{att}'_\kappa(\mathsf{fail},x') \end{array}$$

By simple resolution between these clauses and (RBad1') and (RBad2'), we obtain the clauses

$$\begin{array}{c} \mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_\kappa(x,y') \wedge x' \neq y' \to \mathrm{bad} \\ \mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_\kappa(y,x') \wedge x \neq y \to \mathrm{bad} \end{array}$$

We can be more general by considering two attacker facts that are not in the same phase thanks to the rule (Rap'). Hence we have the following two rules:

$$\mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_{\kappa'}(x,y') \wedge x' \neq y' \to \mathrm{bad} \quad \text{(REq1)}$$

$$\mathrm{att}'_\kappa(x,x') \wedge \mathrm{att}'_{\kappa'}(y,x') \wedge x \neq y \to \mathrm{bad} \quad \text{(REq2)}$$

These clauses allow us to apply the following simplification rules.

$$\frac{\mathbb{C} \cup \{R = (\mathrm{att}'_\kappa(M,N) \wedge \mathrm{att}'_{\kappa'}(M,N') \wedge H \to C)\} \qquad R \neq \text{(REq1)}}{\mathbb{C} \cup \{\mathrm{att}'_\kappa(M,N) \wedge \mathrm{att}'_{\kappa'}(M,N') \wedge H \wedge (N = N') \to C\}} \ (\mathrm{Eq}_1)$$

$$\frac{\mathbb{C} \cup \{R = (\mathrm{att}'_\kappa(N,M) \wedge \mathrm{att}'_{\kappa'}(N',M) \wedge H \to C)\} \qquad R \neq \text{(REq2)}}{\mathbb{C} \cup \{\mathrm{att}'_\kappa(N,M) \wedge \mathrm{att}'_{\kappa'}(N',M) \wedge H \wedge (N = N') \to C\}} \ (\mathrm{Eq}_2)$$

Indeed, if a derivation of bad exists where the last clause is R and $N \neq N'$, then we can replace the application of R with (REq1) or (REq2) and derive bad immediately. Therefore, we can restrict R to apply only when $N = N'$.

5.4 Natural numbers

As mentioned in Section 4, formulae now contain predicates on natural numbers, such as $M \geq N$, $\neg \mathit{nat}(M)$ and $\mathit{nat}(M)$. The simplification rules presented in this section aim to determine whether they are satisfiable or if some equalities are *forced* by these predicates. Following the work in [CCT18], we rely on the algorithm of Pratt [Pra77] for checking the satisfiability of inequalities.

Proposition 2 ([CCT18]). *There is a polynomial time algorithm* checkeq *that given a conjunction ϕ of inequalities between terms returns:*

- $\bot$ *if ϕ has no solution*
- *a substitution σ' such that for all solutions σ of ϕ, there exists a substitution δ such that $\sigma = \sigma'\delta$.*

Relying on the algorithm checkeq, we can consider the following simplification rules specific for natural number predicates.

$$\frac{\mathbb{C} \cup \{H \wedge \phi \to C\} \qquad \phi = \bigwedge_i M_i \geq N_i \qquad \mathsf{checkeq}(\phi) = \sigma}{\mathbb{C} \cup \{H\sigma \wedge \phi\sigma \to C\sigma\}}$$

$$\frac{\mathbb{C} \cup \{H \wedge \phi \to C\} \qquad \phi = \bigwedge_i M_i \geq N_i \qquad \mathsf{checkeq}(\phi) = \bot}{\mathbb{C}}$$

In [CCT18], the negation of $M \geq M'$ is directly translated into $M' \geq M + 1$. This is only possible because they consider a semi-typed attacker differentiating terms of type natural number from other terms. As such, in a process $\mathsf{in}(c, x : nat); P$, x can only be instantiated by a natural number. In this work however, we consider a fully untyped attacker. Therefore, in our setting, the semantics of $M \geq M'$ is that *both M and M' are natural number and M is greater than M'*. Thus, the negation of $M \geq M'$ is $\neg nat(M) \vee \neg nat(M') \vee M' \geq M + 1$. The following transformations handle the predicates $\neg nat(M)$, $nat(M)$ and check that all terms M, M' in an inequality $M \geq M'$ can be natural numbers.

$$\frac{\mathbb{C} \cup \{M \geq M' \wedge H \to C\}}{\mathbb{C} \cup \{nat(M) \wedge nat(M') \wedge M \geq M' \wedge H \to C\}}$$

$$\frac{\mathbb{C} \cup \{nat(succ(M)) \wedge H \to C\}}{\mathbb{C} \cup \{nat(M) \wedge H \to C\}}$$

$$\frac{\mathbb{C} \cup \{\neg nat(succ(M)) \wedge H \to C\}}{\mathbb{C} \cup \{\neg nat(M) \wedge H \to C\}}$$

$$\frac{\mathbb{C} \cup \{nat(f(M_1, \ldots, M_n)) \wedge H \to C\} \qquad f \notin \{succ, zero\}}{\mathbb{C}}$$

$$\frac{\mathbb{C} \cup \{\neg nat(f(M_1, \ldots, M_n)) \wedge H \to C\} \qquad f \notin \{succ, zero\}}{\mathbb{C} \cup \{H \to C\}}$$

$$\frac{\mathbb{C} \cup \{\neg nat(M) \wedge nat(M) \wedge H \to C\}}{\mathbb{C}}$$

It is easy to see that these transformations preserve the solutions of a formulae. In the rest of this paper, we will regroup all these transformations into a single transformation rule, called Nat, which typically corresponds to the application of these rules on a formula of a clause until either reaching a fixpoint or until removing the clause from the set (when the formula has no solution).

Specific rules for correspondence properties By definition, an attacker knows all natural numbers since the function symbols *zero* and *succ* are public. In particular, the following clauses can be found in $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$:

$$\begin{array}{rl} \mathrm{att}_\kappa(x) \to \mathrm{att}_\kappa(succ(x)) & \text{(R+)} \\ \to \mathrm{att}_\kappa(zero) & \text{(R0)} \end{array}$$

Hence, when the resolution generates a clause $H \wedge \phi \to \mathrm{att}_\kappa(M)$ where M is a natural number, we can ignore this clause. This is sound since we can always build a derivation of $\mathrm{att}_\kappa(M)$ from the clauses (R+) and (R0). Note that this derivation satisfies any trace from $\mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)$ as they require the attacker to always build the natural number used in the trace at its beginning.

The rule is thus formalised as follows:

$$\frac{\mathbb{C} \cup \{R = (H \wedge \phi \to \mathrm{att}_\kappa(M))\} \qquad \phi \models nat(M) \qquad R \notin \{(\text{R+}), (\text{R0})\}}{\mathbb{C}} \text{ (NatCl)}$$

Specific rules for equivalence properties As in the correspondence properties, the following clauses are in $\mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$:

$$\begin{array}{rl} \mathrm{att}'_\kappa(x, y) \to \mathrm{att}_\kappa(succ(x), succ(y)) & \text{(R+')} \\ \to \mathrm{att}'_\kappa(zero, zero) & \text{(R0')} \end{array}$$

Thus we can once again apply a simplification rule when the conclusion of a clause is a natural number.

$$\frac{\mathbb{C} \cup \{R = (H \wedge \phi \to \mathrm{att}'_\kappa(M, M))\} \qquad \phi \models nat(M) \qquad R \notin \{(\text{R+'}), (\text{R0'})\}}{\mathbb{C}} \text{ (NatCl')}$$

Additionally when we obtain a clause $R = (\mathrm{att}'_\kappa(M, N) \wedge H \to C)$ where M or N is a natural number, we can enforce that $M = N$. Indeed, we know from Definition 12 that we only consider traces where the attacker build the natural number in the trace directly at its beginning. Thus, intuitively, if a derivation relies on the clause R, then it would imply that $T, \tau \vdash^{\kappa_{io}}_{IO} \mathrm{att}'_\kappa(M, N)$ for some step τ and that there is a derivation $\mathcal{D}$ of $\mathrm{att}'_\kappa(M, N)$. By a minimality argument, we can assume that $\mathcal{D}$ does not rely on the clause R. (Otherwise, there is a smaller derivation $\mathcal{D}'$ of $\mathrm{att}'_\kappa(M, N)$ within $\mathcal{D}$.) Assuming that $M \in \mathbb{N}$ and $M \neq N$, we would therefore be able to build a derivation of bad using $\mathcal{D}$, the clause (REq1) and the derivation of $\mathrm{att}'_\kappa(M, M)$ using the clauses (R+') and (R0'). Note that this derivation of bad does not use the clause R.

$$\frac{\mathbb{C} \cup \{\mathrm{att}'_\kappa(M, N) \wedge H \wedge \phi \to C\} \qquad \phi \models nat(M) \vee nat(N)}{\mathbb{C} \cup \{\mathrm{att}'_\kappa(M, N) \wedge H \wedge \phi \wedge (M = N) \to C\}} \text{ (NatHyp')}$$

5.5 Applying ProVerif lemmas

As mentioned in Section 2.4, we introduce in this paper ProVerif lemmas that are correspondence queries $F_1 \wedge \ldots \wedge F_n \Rightarrow \psi$ that do not contain any injective event nor nested query. These lemmas are typically declared and proved before the current queries. Thanks to Lemma 1, we know that all queries can be transformed into queries in disjunctive normal

form (DNF). Hence, we assume in this section that all lemmas are in DNF. In this section, we will denote $\mathcal{L}, \mathcal{L}', \ldots$ the sets of lemmas.

Concretely, given an initial instrumented configuration $\mathcal{C}_I$, an integer κ_{io}, a set of ProVerif lemmas $\mathcal{L}$ and a IO-κ_{io}-compliant query ϱ, ProVerif will first try prove that $\mathcal{C}_I$ satisfies all lemmas in $\mathcal{L}$ without any assumption, then it will prove that $\mathcal{C}_I$ satisfies ϱ using the proved lemmas in the saturation. In that respect, ProVerif will thus saturate twice the main set of clauses $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$, once to prove the lemmas in $\mathcal{L}$ and a second time to prove the query ϱ but using the simplification rules we describe in this subsection.

In Lemma 1, we showed that for all predicate sets $\mathcal{S}_p$, for all executable facts F in a trace T at step τ, there exists a derivation $\mathcal{D}$ of F such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$. In particular, $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$ indicates that for all facts F' in $\mathcal{D}$, if $pred(F') \in \mathcal{S}_p$ then $T, \tau' \vdash^{\kappa_{io}}_{IO} F'$ for some τ'. Thus, given a lemma $F_1 \wedge \ldots \wedge F_n \Rightarrow \bigvee_{i=1}^{m} \psi_i$, if there exists a substitution σ such that $F_1\sigma, \ldots, F_n\sigma$ are facts of $\mathcal{D}$ with allowed predicates (i.e. predicates in $\mathcal{S}_p$) then it implies that there exists $i \in \{1, \ldots, m\}$ such that $\psi_i\sigma$ holds on the trace T.

This property is the core of the simplification rule below. For this rule, given a conjunction of atomic formulas ψ, we denote by $\lceil \psi \rceil^{sure}$ the formula obtained from ψ by replacing all events event(ev) with sure-events s-event(x, ev) with x a fresh variable and all bievents event$'(ev, ev')$ with sure-events s-event$'(ev, ev')$. Similarly, given a query conclusion ψ, we denote by $\lceil \psi \rceil^{may}$ the query conclusion obtained from ψ by replacing all events event(ev) with may-events m-event(x, ev) with x a fresh variable and all events event$'(ev, ev')$ with may-events m-event$'(ev, ev,')$. Moreover, we denote by $\mathbb{C}_{std}$ the set of clauses containing for all phases the clauses (Rl), (Rap), (Rl'), (Rap'), (Rf$_g$), (Rf$'_g$), (Rf$'_{\pi_i^g}$) and (Rf$_{\pi_i^g}$) for all $g \in \mathcal{F}_{data}$ and i. Note that since the function symbols *succ* and *zero* are data constructors, the set $\mathbb{C}_{std}$ contains the clauses for natural numbers (R+) and (R0).

$$\frac{\begin{array}{c}\mathbb{C} \cup \{R = (H \to C)\} \qquad (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j) \in \mathcal{L} \qquad R \notin \mathbb{C}_{std} \\ \forall i, pred(F_i) \in \mathcal{S}_p \text{ and either } \lceil F_i\sigma \rceil^{sure} \in H \\ \text{or } (\lceil F_i\sigma \rceil^{may} = C \text{ and } \forall j, \forall F \in \psi_j,\ mgu(\lceil F\sigma \rceil^{may}, C) = \bot)\end{array}}{\mathbb{C} \cup \{H \wedge \lceil \psi_j\sigma \rceil^{sure} \to C\}_{j=1}^{m}} \text{(Lem}(\mathcal{L}, \mathcal{S}_p))$$

Notice that we require $\lceil F_i\sigma \rceil^{sure} \in H$ and not $F_i\sigma \in H$. Indeed, events in lemmas are expressed using the predicate event whereas events in the hypotheses of a clause are expressed using the predicate s-event. Similarly, when matching a the premise of the lemma with the conclusion C, we require that $\lceil F_i\sigma \rceil^{may} = C$ and not $F_i\sigma$. Notice that in this case, we also require the conclusion C to not be unifiable with any of the events in the conclusion of lemma $\bigvee_{j=1}^{m} \psi_j$. The reason for this condition is that we want to ensure that any events in the hypotheses of a clause occurs strictly before the conclusion of the clause. However, a lemma only guarantee us that the events in $\bigvee_{j=1}^{m} \psi_j$ occur before or at the same time as the fact F_i. Hence, by requiring that the conclusion C is not unifiable with any events in $\bigvee_{j=1}^{m} \psi_j$, we ensure that they occur strictly before C.

Finally, notice that we prevent lemmas to be applied on clauses in $\mathbb{C}_{std}$: these clauses are useful to guarantee some properties of the resolution algorithm, and so should not be modified by applying lemmas. Applying lemmas to these clauses is also unlikely to bring useful information during the saturation procedure.

Applying inductive lemmas The rule Lem$(\mathcal{L}, \mathcal{S}_p)$ can only be applied when ProVerif already proved that the lemmas in $\mathcal{L}$ are true. For proving an inductive lemma, we can in fact

use a similar simplification rule. As mentioned in Section 3.3, we prove queries by induction on the size of the trace and the multiset of the steps on which the hypotheses of the query are satisfied.

As for the application of non-inductive lemmas, we know from Theorem 1 that for all predicate sets $\mathcal{S}_p$, for all executable facts F in a trace T at step τ, there exists a derivation $\mathcal{D}$ of F such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$. Hence by Definition 18, for all nodes of the derivation, if the rule labeled on the node is not the rule (Rl) then all facts on the outgoing edges of the node (i.e. its hypotheses) are satisfied by the trace at a step strictly before the fact on the incoming edge (i.e. its conclusion), provided they have predicates allowed by $\mathcal{S}_p$. Moreover, Item 4 of Definition 18 indicates typically that when a fact $\mathrm{att}_\kappa(f(M_1, \ldots, M_n))$ with $f \in \mathcal{F}_{data}$ is in the hypotheses of a rule then either the rule is a projection rule or $T, \tau_0 \vdash^{\kappa_{io}}_{IO} F$ when $pred(F) \in \mathcal{S}_p$. Therefore, we can apply on these facts our inductive hypothesis. Note that even if the conclusion of the clause may not have an allowed predicate, the main derivation will necessarily have allowed predicates in its conclusion since we are proving the query by induction.

$$\frac{\begin{array}{c}\mathbb{C} \cup \{R = (H \to C)\} \qquad R \notin \mathbb{C}_{std} \\ (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j) \in \mathcal{L} \qquad \forall i, pred(F_i) \in \mathcal{S}_p \text{ and } \lceil F_i\sigma \rceil^{sure} \in H\end{array}}{\mathbb{C} \cup \{H \wedge \lceil \psi_j\sigma \rceil^{sure} \to C\}_{j=1}^{m}} \,(\mathrm{Ind}(\mathcal{L}, \mathcal{S}_p))$$

5.6 The saturation procedures

We combine the simplification rules to define our saturation procedures for both correspondence and equivalence queries. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. Intuitively, $\mathcal{L}$ is the set of lemmas that was already proved whereas $\mathcal{L}_i$ is the set of inductive lemma obtained from queries that need to be proved by induction. Let $\mathbb{C}$ be a set of clauses. Let $\mathcal{S}_p$ a set of predicates. We define the algorithm $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ as the repeated application on $\mathbb{C}$ of the rules Taut, Red, Att, Att', R_ϕ, DataHyp', DataHyp, DataCl, DataCl', Nat, NatCl, NatCl', and the rules $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$, $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$ until a fixpoint is reached.

We say that a clause is *simplified* when it is left unchanged by the rules used in $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ other than the rules $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$ and $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$. As such, all clauses in $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ are simplified.

Note that $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ includes the application of the rules on clauses and biclauses. However, the syntax of clauses and biclauses being distinct, in pratice a rule for biclauses will never be applied when proving a correspondence query and vice-versa.

We also define the function $\mathsf{condense}_{\mathcal{S}_p}(\mathbb{C})$ that eliminates from $\mathbb{C}$ the clauses that are subsumed by other clauses from $\mathbb{C}$ and that applies the rule $\mathrm{GRed}(\mathcal{S}_p)$ repeatedly until a fixpoint is reached.

Finally, we define the algorithm $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ that (i) applies $\mathsf{condense}_{\mathcal{S}_p}(\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}))$, (ii) generates a new clause R by application of the resolution rule Res and simplifies it hence generating a new set of clauses $\mathbb{C}' = \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$, (iii) condenses the set $\mathbb{C}'$ with the current set of clauses, (iv) repeats the steps (ii) and (iii) until a fixpoint is reached, (v) returns the set of clauses R such that $\mathrm{sel}(R) = \emptyset$.

5.7 Soundness of the saturation procedures

The soundness of the saturation procedure holds for all sets of clauses that satisfy some origination property defined as follows.

Definition 21. *We say that a clause $H \to C$ is well originated when*

- *for all variables x (different from a session identifier variable), if x occurs in C (resp. $\mathsf{fst}(C)$, $\mathsf{snd}(C)$) or in a fact F (resp. $\mathsf{fst}(F)$, $\mathsf{snd}(F)$) of H different from an event (resp. bievent) then there exist F' in H and a term M such that x occurs in M and either $F' = \mathrm{att}_\kappa(M)$ (resp. $\mathsf{fst}(F'), snd(F')$) or $F' = \mathrm{att}_\kappa(N, M)$ (resp. $\mathsf{fst}(F'), snd(F')$) for some N or $F' = \mathrm{table}_\kappa(M)$ (resp. $\mathsf{fst}(F'), \mathsf{snd}(F')$).*
- *if C is a fact with phase κ then for all facts in H with phase κ', we have $\kappa \geq \kappa'$.*

A set of clauses is well originated when all its clauses are well originated.

In terms of processes, the well-origination property states that all variables in the clauses have always been introduced by an input or a table lookup. Moreover, it also states that phases can never decrease. All clauses generated by ProVerif are in fact well-originated.

A mentioned in Section 3.3, ProVerif is now capable of proving queries by induction and can rely on lemmas. Therefore, our soundness results also exhibit the inductive hypotheses and lemmas. We gather these properties in the predicate $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ defined below.

Definition 22. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. Let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant restrictions. Let $\tilde{\tau}$ be a tuple of steps. Let $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)_{|\mathcal{R}}$. We define the predicate $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ to hold if and only if*

- *for all $\varrho \in \mathcal{L}$, $(\vdash^{\kappa_{io}}_{IO}, \vdash_i, \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)_{|\mathcal{R}}) \models \varrho$;*
- *for all $\varrho \in \mathcal{L}_i$, for all $T' \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)_{|\mathcal{R}}$, for all tuples of steps $\tilde{\tau}'$, if $(T', \tilde{\tau}') <_{ind} (T, \tilde{\tau})$ then $\mathcal{IH}_\varrho(T', \tilde{\tau}')$.*

We can now state the soundness of the saturation procedures.

Theorem 2. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. Let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant restrictions. Let $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)_{|\mathcal{R}}$.*

For all well originated sets of clauses $\mathbb{C}$ containing $\mathbb{C}_{std}$, for all derivations $\mathcal{D}$ of F at step τ from $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, (\tau))$ and $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$, there exists a derivation $\mathcal{D}'$ of F at step τ from $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\mathbb{C}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}'$.

In Lemma 12, we showed a soundness result for correspondence queries on bitraces by focusing only on the set of the clauses $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io})$ which does not contain clauses with the fact bad as conclusion. Hence, the natural next step would have been to saturate only the set $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io})$. However, our experiment showed that on many examples, the saturation would enter a loop and never terminate whereas the saturation on the full set of clauses $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$ would terminate. At first glance, this result may seem counter intuitive as $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io})$ is strictly included in $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$. However, one of the main reasons for this behaviour comes from our definition of subsumption (Definition 16) in which a clause with bad as conclusion can subsume a clause with a conclusion different from a fact bad.

This definition of subsumption is tailored for equivalence properties as it indicates that we can remove a clause if we are already able to derive bad from its hypotheses. However, when removing all clauses with bad as conclusion, clauses that would have been removed are now kept in the saturation which leads to the termination issues we experimented. To circumvent such problems, we allow some clauses with bad as conclusion to be saturated. Next theorem shows that we can consider a flexible set of clauses to saturate. Once again, we consider the predicate $\mathcal{H}yp'_{\mathcal{L},\mathcal{L}_i}(T,\tilde{\tau})$ for the bitrace T and tuple of steps $\tilde{\tau}$.

Theorem 3. *Let $\mathcal{C}_I$ be an initial instrumented biconfiguration. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of* ProVerif *IO-κ_{io}-compliant lemmas on bitraces. Let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant bi-restrictions. Let $T \in \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|\mathcal{R}}$.*

For all well originated sets of clauses $\mathbb{C}$ containing $\mathbb{C}'_{std}$, for all derivations $\mathcal{D}$ of F (F possibly being bad*) at step τ from $\mathbb{C} \cup \mathbb{C}'_e(T)$ such that $\mathcal{H}yp'_{\mathcal{L},\mathcal{L}_i}(T,(\tau))$ and $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}$, there exists a derivation $\mathcal{D}'$ of either F or* bad *at step τ from $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\mathbb{C}) \cup \mathbb{C}'_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}'$.*

Since we added possibly some clauses with bad as conclusion, Theorem 3 indicates that we may not be able to build a derivation of the fact F but rather a derivation of bad. In such a case, ProVerif cannot prove the correspondence query. Though this is a limitation w.r.t. the correspondence query, this is still satisfactory in the context of equivalence queries. Indeed, correspondence queries on bitraces are introduced in ProVerif only as lemmas for a proof of equivalence. Hence, if the saturation for the correspondence query yields bad to be derivable, we can deduce that the lemma would not be sufficient to help proving the equivalence query itself (as it would most probably also yield a derivation of bad).

In the implementation ProVerif, we had to decide how to choose $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io}) \subseteq \mathbb{C} \subseteq \mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$. Taking $\mathbb{C}$ too close to $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$ would help terminating but would often derive bad. On other hand, taking $\mathbb{C}$ too close to $\mathbb{C}^c(\mathcal{C}_I, \kappa_{io})$ would often lead the saturation procedure into a loop. Experimentation showed that a good tradeoff would be to remove all clauses with bad as conclusion in $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io})$ but to keep all the clauses from $\mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$. Intuitively, it means that we let the attacker distinguish processes from its knowledge while preventing him from distinguishing the processes through their control flow.

Remark 12. Lemmas on bitraces are best proved by induction since the inductive hypothesis is also applied during the saturation procedure. This increases the chances to avoid obtaining a derivation of bad. ▶

6 The solving procedure

The saturation procedures described in section 5 provide a simplified set of clauses such that all executable facts are derivable from this set of clauses. In this section, we describe how we verify that a correspondence or equivalence query holds on the set of saturated clauses.

6.1 The conjunction predicate

The correspondence queries we aim to verify are of the form $F_1 \wedge \ldots \wedge F_n \Rightarrow \psi$ meaning that we need to find derivations for instantiations of $F_1, \ldots, F_n$. However, the saturation only provides us with clauses $H \rightarrow C$ where C is a single fact. Therefore, we introduce a new predicate that will represent the conjunction of facts $F_1 \wedge \ldots \wedge F_n$.

Definition 23. *Consider a sequence of predicates $seq = [p_1, \ldots, p_n]$ with $i_1, \ldots, i_n$ their respective arity. We define the predicate $conj_{seq}$ as a predicate of arity $i_1 + \ldots + i_n$. The conjunction of facts $\bigwedge_{k=1}^{n} p_k(pt_1^k, \ldots, pt_{i_k}^k)$ is then represented by the following conjunction fact F:*

$$F = conj_{seq}(pt_1^1, \ldots, pt_{i_1}^1, pt_1^2, \ldots, pt_1^n, \ldots, pt_{i_n}^n)$$

Intuitively, the conjunction fact F regroups all the arguments of all facts $p_k(pt_1^k, \ldots, pt_{i_k}^k)$ in its own arguments.

Example 19. Let $seq = [\text{m-event}, \text{m-event}]$. The clause $\text{s-event}(o_1[], A) \rightarrow conj_{seq}(o_2[], B, o_3[], C)$ represents that when both events B and C are executed respectively at occurrence $o_2[]$ and $o_3[]$ then the event A was previously executed at occurrence $o_1[]$.

Note that it is important for *seq* to be a sequence and not a set. By being a sequence, we can retrieve from a conjunction fact the corresponding conjunction of facts. ▶

For sake of simplicity, we will now denote by $\bigwedge\!\!\!\!\bigwedge_{i=1}^{n} F_i$ the conjunction fact representing $\bigwedge_{i=1}^{n} F_i$.

Remark 13. Note that in the rest of this section, even if the sequence *seq* contains only one predicate, we will still rely on the predicate $conj_{seq}$ for the solving procedure. ▶

6.2 Ordered clauses and derivations

Given a query $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi$, we use the conjunction predicate $\bigwedge\!\!\!\!\bigwedge_{i=1}^{n} F_i$ to generate a set of clauses that represents any executable instantiation of $\bigwedge_{i=1}^{n} F_i$ and that we compute from the set of saturated clauses $\mathbb{C}_{sat}$ obtained with $\mathsf{saturate}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\mathbb{C})$ as described in the previous section.

Intuitively, we will saturate the set of clauses $\mathbb{C}_{sat} \cup \{\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may} \rightarrow \bigwedge\!\!\!\!\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may}\}$. At the end of the saturation, the set of clauses with no selectable hypothesis and whose conclusion is an instance of $\bigwedge\!\!\!\!\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may}$ will represent the set of possible executable instantiations of $\bigwedge_{i=1}^{n} F_i$.

We cannot however reuse exactly the saturation procedure $\mathsf{saturate}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\mathbb{C})$ due to the transformation rules for proving ProVerif lemmas by induction. Indeed, these rules rely on the fact that we only consider derivations of a fact that satisfy $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$. In particular, an inductive lemma is applied during the saturation because we know that all facts in the hypotheses occurs strictly before the conclusion. Since we now consider conjunction of facts with the clause $\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may} \rightarrow \bigwedge\!\!\!\!\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may}$, we would like to apply the inductive lemma when the hypotheses of the clauses on which the lemma is applied satisfy the strict inductive order $<_{ind}$ defined in Section 3.3, and in particular the strict order on multisets of steps.

To achieve this, in a clause $H \rightarrow \bigwedge\!\!\!\!\bigwedge_{i=1}^{n} F_i$, we augment each fact F' in H with a partial function that indicates whether the fact occurred before or strictly before each fact F_i of the conclusion. Hence assuming that the facts F_i are satisfied at steps τ_i in the derivation, these partial functions allow us to compute whether some facts $F'_1, \ldots, F'_m$ in H satisfied at step $\tau'_1, \ldots, \tau'_m$ verify the strict order $\{\!\{\tau'_1, \ldots, \tau'_m\}\!\} <_m \{\!\{\tau_1, \ldots, \tau_n\}\!\}$.

Definition 24. *We call* ordering function *a partial function $\delta : \mathbb{N} \rightarrow \{<, \leq\}$. We call* ordered facts*, denoted F^{δ}, a fact F annotated with an ordering function δ. Finally, an* ordered clause *is a clause of the form $H \rightarrow \bigwedge\!\!\!\!\bigwedge_{i=1}^{n} F_i$ where each elements of H are ordered facts.*

Example 20. Consider the ordered clause:

$$\text{s-event}(o_1, ev_1)^{\delta_1} \wedge \text{s-event}(o_2, ev_2)^{\delta_2} \rightarrow \text{m-event}(o_3, ev_3) \curlywedge \text{m-event}(o_4, ev_4)$$

with $\delta_1 = [1 \mapsto \leq; 2 \mapsto <]$ and $\delta_2 = \emptyset$. The clause indicates that event ev_1 occurs before (not necessarily strictly) ev_3, and occurs strictly before ev_4. However, since $\delta_2 = \emptyset$, we have no indication other than ev_2 occurs before ev_3 or ev_4 (not necessarily strictly). ▶

When the partial function δ is not specified in an ordered fact, we assume that $\delta = \emptyset$.

Definition 25. *Let $\mathcal{S}_p$ be a set of predicates. We say that an ordered clause $F_1^{\delta_1} \wedge \ldots \wedge F_n^{\delta_n} \wedge \phi \rightarrow C$ satisfies $\mathcal{S}_p$ when for all $i \in \{1, \ldots, n\}$, if $pred(F_i) \notin \mathcal{S}_p$ then $< \notin img(\delta_i)$.*

We now extend the notion of derivation to take into account ordered clauses.

Definition 26. *Let $\kappa_{io} \in \mathbb{N}$. Let $\mathbb{C}_{sat}$ be a set of selection-free clauses. Let $\mathbb{C}$ be a set of ordered clauses. Let $F = \curlywedge_{i=1}^{n} F_i$ be a closed conjunction fact. Let $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ be a tuple of steps. An* ordered derivation *$\mathcal{D}$ of F at steps $\tilde{\tau}$ from $\mathbb{C}$ and $\mathbb{C}_{sat}$ is a finite tree defined as follows:*

1. *the root is unlabeled; the child of the root is labeled by an ordered clause from $\mathbb{C}$ and all other nodes are labeled by clauses in $\mathbb{C}_{sat}$;*
2. *the incoming edge of the child of the root is labeled by $F, \tilde{\tau}$; all other incoming edges of nodes are labeled by a fact and a step;*
3. *if the child η of the root is labeled by a clause $R = G_1^{\delta_1} \wedge \ldots \wedge G_m^{\delta_m} \wedge \phi \rightarrow C$ then there exists a substitution σ and some steps $\tau_{G_1}, \ldots, \tau_{G_m}$ such that $\vdash_i \phi\sigma$, $C\sigma = F$ and η has m children $\eta_1, \ldots, \eta_m$ where for all $k \in \{1, \ldots, m\}$,*
   (a) *the edge between η and η_k is labeled by $G_k\sigma, \tau_{G_k}$,*
   (b) *the subtree rooted in η_k is a derivation of $G_k\sigma$ at step τ_{G_k}, and*
   (c) *for all $i \in \{1, \ldots, n\}$, if $\delta_k(i)$ is defined then $\tau_{G_k}\ \delta_k(i)\ \tau_i$.*

Let T be a IO-κ_{io}-compliant trace. Let $F = \curlywedge_{i=1}^{n} F_i$ be closed facts and $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ be a tuple of steps. We say that T satisfies an ordered derivation $\mathcal{D}$ of F at step $\tilde{\tau}$ w.r.t. $\mathcal{S}_p$, *denoted $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$ when*

1. *for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash_{IO}^{\kappa_{io}} F_i$;*
2. *if the child η of the root is labeled by a clause $R = G_1^{\delta_1} \wedge \ldots \wedge G_m^{\delta_m} \wedge \phi \rightarrow C$ with $\eta_1, \ldots, \eta_m$ children such that for all $k \in \{1, \ldots, m\}$, $\eta \xrightarrow{G_k\sigma, \tau_{G_k}} \eta_k$ then we have for all $k \in \{1, \ldots, m\}$, the subderivation $\mathcal{D}_k$ rooted in η_k satisfies $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}_k$;*

Intuitively, an ordered derivation is a conjunction of multiple derivations that may share hypotheses and that satisfy the ordering functions of ordered clauses. In particular, item 3c of the definition indicates that when the ordering function $\delta_k(i)$ is defined, i.e. either $\delta_k(i) = <$ or $\delta_k(i) = \leq$, the steps τ_{G_k} and τ_i must satisfies the ordering function, i.e. $\tau_{G_k} < \tau_i$ or $\tau_{G_k} \leq \tau_i$ respectively.

6.3 Ordered transformation rules

The saturation procedure described in Section 5 only takes into account standard clauses and so we need to modify all transformation rules for ordered clauses.

Resolution rule. As mentioned in the beginning of this section, the solving procedure typically aims to saturate the set of clauses $\mathbb{C}_{sat} \cup \{\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may} \to \bigwedge\!\!\!\bigwedge_{i=1}^{n} \lceil F_i \rceil^{may}\}$. Note that all clauses R in $\mathbb{C}_{sat}$ satisfy $\mathrm{sel}(R) = \emptyset$ and no more transformation rules can be applied on them. Hence, we can separate them from the set of clauses we will build through the solving procedure. In particular, we can show a simple invariant that conjunction facts can only occur as a conclusion of a clause. Thus, the application of the resolution rule is necessarily between a clause $H \to C$ in $\mathbb{C}_{sat}$ and an ordered clause $H_o \wedge F^{\delta} \to \bigwedge\!\!\!\bigwedge_{i=1}^{n} F_i$. Recall that Theorem 2 indicates that we are only interested in derivation where all facts of the hypotheses of a standard clause are satisfied strictly before its conclusion unless their predicates is not in $\mathcal{S}_p$. Hence, when applying a resolution between C and F^{δ} it implies that all facts in H are satisfied strictly before F and the ordering function δ can be strictly propagated to the facts in H.

Formally, given an ordering function δ, we denote by $\delta^{<}$ the partial function with the same domain as δ and such that $\delta^{<}(i) = <$ for all $i \in \mathrm{dom}(\delta)$. The resolution rule is therefore defined as follows:

$$\frac{\begin{array}{c} F_1 \wedge \ldots \wedge F_n \to C \in \mathbb{C}_{sat} \qquad F^{\delta} \wedge H_o \to C' \in \mathbb{C} \qquad \sigma = mgu(F, C) \\ \text{for all } i \in \{1, \ldots, n\},\ \delta_i = \delta^{<} \text{ if } pred(F_i) \in \mathcal{S}_p \text{ else } \delta_i = \delta \end{array}}{F_1^{\delta_1}\sigma \wedge \ldots \wedge F_n^{\delta_n}\sigma \wedge H_o\sigma \to C'\sigma} \ (\mathrm{Res}_o(\mathcal{S}_p))$$

Ordered subsumption. We also need to modify the subsumption definition so that it takes into account the ordering functions on facts.

Definition 27. *Given two ordering functions δ and δ', we denote $\delta \sqsupseteq_o \delta'$ if for all $j \in \mathbb{N}$, (i) if $\delta(j)$ is defined then $\delta'(j)$ is defined (ii) if $\delta(j) = <$ then $\delta'(j) = <$.*

Given two multisets H_1 and H_2 of ordered facts, we denote $H_1 \subseteq_o H_2$ if $H_1 = \{\!\{F_1^{\delta_1}, \ldots, F_n^{\delta_n}\}\!\}$, $H_2 = \{\!\{F_1^{\delta'_1}, \ldots, F_n^{\delta'_n}\}\!\} \cup H'_2$ and for all $i \in \{1, \ldots, n\}$, $\delta_i \sqsupseteq_o \delta'_i$.

Let $H_1 \wedge \phi_1 \to C_1$ and $H_2 \wedge \phi_2 \to C_2$ two ordered clauses. We say that $H_1 \wedge \phi_1 \to C_1$ subsumes $H_2 \wedge \phi_2 \to C_2$, denoted $(H_1 \wedge \phi_1 \to C_1) \sqsupseteq_o (H_2 \wedge \phi_2 \to C_2)$, when there exists σ such that (i) either $C_1\sigma = C_2$ or $C_1 = \mathrm{bad}$ (ii) $H_1\sigma \subseteq_o H_2$ (iii) $\phi_2 \models \phi_1\sigma$.

Note that Definition 27 only differs from the original definition of subsumption (Definition 16) on the notion of multiset inclusion $\subseteq_o$ of ordered facts . Typically, we need to ensure that the subsumed ordered clause $H_2 \wedge \phi_2 \to C_2$ has stronger ordering functions w.r.t. the subsuming clause $H_1 \wedge \phi_1 \to C_1$. In particular, the subsumption of ordering function satisfies the following property.

Lemma 13. *Let δ, δ' two ordering functions such that $\delta \sqsupseteq_o \delta'$. For all steps τ, τ', for all $i \in \mathrm{dom}(\delta)$, if $\tau\ \delta'(i)\ \tau'$ then $\tau\ \delta(i)\ \tau'$.*

Proof. Let τ, τ' two steps and $i \in \mathrm{dom}(\delta)$ such that $\tau\ \delta'(i)\ \tau'$. Since $i \in \mathrm{dom}(\delta)$, then either $\delta(i) = <$ or $\delta(i) = \leq$. In the former case, we deduce from $\delta \sqsupseteq_o \delta'$ that $\delta'(i) = < = \delta(i)$. Hence $\tau\ \delta'(i)\ \tau'$ implies $\tau\ \delta(i)\ \tau'$. In the latter case, we know that $\delta'(i) \in \{<; \leq\}$. Hence $\tau\ \delta'(i)\ \tau'$ implies $\tau \leq \tau'$ and so $\tau\ \delta(i)\ \tau'$. □

Example 21. The following holds:

- The empty order function subsumes all ordering functions, i.e. $\emptyset \sqsupseteq_o \delta$ for all δ.
- $[1 \mapsto \leq] \sqsupseteq_o [1 \mapsto <]$
- $[1 \mapsto \leq; 2 \mapsto \leq] \not\sqsupseteq_o [1 \mapsto <]$

Consider now the ordered clause $R = \text{att}(a[]) \wedge \text{s-event}(o[], A(a[])) \rightarrow \text{att}(h(a[]))$. We have:

- $\text{att}(x) \rightarrow \text{att}(h(x)) \sqsupseteq_o R$
- but $\text{att}(x)^{[1 \mapsto <]} \rightarrow \text{att}(h(x)) \not\sqsupseteq_o R$

The $\text{att}(x)^{[1 \mapsto <]} \rightarrow \text{att}(h(x))$ does not subsume R since we have a stricter condition on the predicate $\text{att}(x)$, i.e. that it does appear strictly before $\text{att}(h(x))$. ▶

Classic simplification rules. Amongst the classical simplification rules, we only need to focus on the rule Red and DataHyp. The other rules remaining as they are (other than they are applied on ordered clauses rather than clauses. In particular they preserve the ordering functions). Note that the rule DataCl will in fact never be applied since the conclusion of an ordered clause is always a conjunction fact and so not an attacker fact.

For the rule Red, when an ordered fact $F_i'^{\delta_i'}$ is a duplicate after instantiation of another fact $F_i^{\delta_i}$ in the hypotheses of the ordered clause (i.e. $F_i'\sigma = F_i$) and the ordering function δ_i' subsumes δ_i (i.e. $\delta_i' \sqsupseteq_o \delta_i$) then we can keep the ordered fact $F_i^{\delta_i}$ and remove $F_i'^{\delta_i'}$ since any derivation of an instance of $F_i^{\delta_i}$ would also be a derivation of an instance of $F_i'^{\delta_i'}$ satisfying the ordering function. Note that when $F_i = F_i'$, we don't need to require $\delta_i' \sqsupseteq_o \delta_i$ and we can consider the *intersection* of ordering functions δ_i and δ_i' which intuitively corresponds to the conjunction of constraints given by δ_i and δ_i'.

Formally, $\delta_i \cap \delta_i'$ is the ordering function defined on the union of domains of δ_i, δ_i' and such that for all $j \in \text{dom}(\delta_i \cap \delta_i')$, $(\delta_i \cap \delta_i')(j) = <$ when $< \in \{\delta_i(j), \delta_i'(j)\}$ and $(\delta_i \cap \delta_i')(j) = \leq$ otherwise. Intuitively, if τ and τ' are the steps on which F_i and F_i' are satisfied then it implies that τ and τ' satisfy the ordering function δ_i and δ_i' respectively. Notice that in such a case, $\min(\tau, \tau')$ necessarily satisfies both δ_i and δ_i', so satisfies $\delta_1 \cap \delta_2$. Hence, we can keep only the derivation corresponding to the step $\min(\tau, \tau')$.

For the rule DataHyp, we argued that this transformation rule was corresponding intuitively to an application of the resolution rule with the clause $(\text{Rf}_g) = \text{att}_\kappa(x_1) \wedge \ldots \text{att}_n(x_n) \rightarrow \text{att}_\kappa(g(x_1, \ldots, x_n))$. Note that $\text{sel}(\text{Rf}_g) = \emptyset$ since the facts $\text{att}_\kappa(x_i)$ are not selectable. Hence $(\text{Rf}_g) \in \mathbb{C}_{sat}$ and so the same argument is valid for ordered clauses.

$$\frac{\begin{array}{c}\mathbb{C} \cup \{H \wedge \phi \wedge \bigwedge_{i=1}^n F_i^{\delta_i} \wedge F_i'^{\delta_i'} \rightarrow C\} \quad \forall i, F_i'\sigma = F_i \text{ and } (F_i' = F_i \text{ or } \delta_i' \sqsupseteq_o \delta_i) \\ \phi \models \phi\sigma \qquad \text{dom}(\sigma) \cap fv(H, C, F_1, \ldots, F_n) = \emptyset\end{array}}{\mathbb{C} \cup \{H \wedge \phi\sigma \wedge \bigwedge_{i=1}^n F_i^{\delta_i \cup \delta_i'} \rightarrow C\}} \ (\text{Red}_o)$$

$$\frac{\begin{array}{c}\mathbb{C} \cup \{\text{att}_\kappa(g(M_1, \ldots, M_m))^\delta \wedge H \rightarrow C\} \quad g \in \mathcal{F}_{data} \\ \text{if } \text{att}_\kappa \in \mathcal{S}_p \text{ then } \delta' = \delta^< \text{ else } \delta' = \delta\end{array}}{\mathbb{C} \cup \{\text{att}_\kappa(M_1)^{\delta'} \wedge \ldots \wedge \text{att}_\kappa(M_m)^{\delta'} \wedge H \rightarrow C\}} \ (\text{DataHyp}_o)$$

As usual, we have a version of the transformation rules for bitraces that we denote DataHyp'$_o$ where bifacts $\mathrm{att}'_\kappa(g(M_1,\ldots,M_m), g(N_1,\ldots,N_m))$ are split into $\mathrm{att}'_\kappa(M_1,N_1),\ldots,$ $\mathrm{att}'_\kappa(M_m,N_m)$.

General redundancy In order for the general redundancy rule to take into account ordering functions, we need to amend the notion of partial derivation. First, as we need to consider a derivation of a conjunction fact, only the rule labeling the child of the root will be an ordered clause and all the clauses labeling other nodes are standard clauses from $\mathbb{C}_{sat}$. Second, all edges are labeled by ordered facts instead of just facts. Third, the ordering functions of the ordered facts must satisfy the ordering functions of the ordered clause. The formal definition is given below.

Definition 28. *Let $\mathbb{C}_{sat}$ be a set of clauses. Let $\mathbb{C}$ be a set of ordered clauses. Let $\mathcal{S}_p$ be a set of predicates. Let F be a conjunction fact (not necessarily closed). A* partial ordered derivation *$\mathcal{D}$ of F from $\mathbb{C}$ and $\mathbb{C}_{sat}$ w.r.t. $\mathcal{S}_p$ is a finite tree defined as follows:*

- *The root is not labeled and has one outgoing edge labeled F.*
- *The child of the root is labeled by an ordered clause $F_1^{\delta_1} \wedge \ldots \wedge F_n^{\delta_n} \wedge \phi \to C$ from $\mathbb{C}$ and has n outgoing subderivation $\mathcal{D}_p^1,\ldots,\mathcal{D}_p^n$ and one outgoing unlabeled leave whose edge is labeled ϕ.*
- *There exists a substitution σ such that $C\sigma = F$ and for $i = 1\ldots n$, $\mathcal{D}_p^i$ is a partial derivation of $F_i\sigma$ from $\mathbb{C}_{sat}$.*

For all $i \in \{1,\ldots,n\}$, we define the multiset $\mathbb{F}_i$ of formulas and ordered facts as follows:

- *if D_p^i contains only a root and an unlabeled leaf, i.e. $\mathbb{F}_{us}(D_p^i) = \{\!\{F_i\sigma\}\!\}$ then $\mathbb{F}_i = \{\!\{F_i\sigma^{\delta_i}\}\!\}$*
- *else $\mathbb{F}_{us}(D_p^i)$ is some multiset $\{\!\{F_{1,i},\ldots,F_{n_i,i},\phi_i\}\!\}$ and $\mathbb{F}_i = \{\!\{F_{1,i}^{\delta_{F_{1,i}}},\ldots,F_{n_i,i}^{\delta_{F_{n_i,i}}},\phi_i\}\!\}$ where for all $j \in \{1,\ldots,n_i\}$, if $pred(F_{j,i}) \in \mathcal{S}_p$ then $\delta_{F_{j,i}} = \delta_i^<$ else $\delta_{F_{j,i}} = \delta_i$.*

Finally, we denote by $\mathbb{F}_s(\mathcal{D})$ the set of all facts labeling the incoming edges of labeled nodes in $\mathcal{D}$, and we denote by $\mathbb{F}_{us}(\mathcal{D})$ the multiset $\{\!\{\phi\sigma\}\!\} \cup \bigcup_{i=1}^n \mathbb{F}_i$.

The general redundancy rule thus becomes:

$$\frac{\begin{array}{c}\mathbb{C} \cup \{H \to F\} \qquad \mathrm{sel}(H \to F) = \emptyset \\ \exists\ \mathcal{D} \text{ partial derivation of } F \text{ from } \mathbb{C} \text{ and } \mathbb{C}_{sat} \text{ w.r.t. } \mathcal{S}_p \text{ such that} \\ \mathbb{F}_{us}(\mathcal{D}) \to F \sqsupseteq_o H \to F \text{ and } pred(\mathbb{F}_s(\mathcal{D})) \cap \mathcal{S}_p = \emptyset\end{array}}{\mathbb{C}}\ (\mathrm{GRed}_o(\mathcal{S}_p))$$

Natural numbers All other rules for dealing with natural numbers remain as they are and preserve the ordering functions.

Application of lemmas Intuitively, the application conditions for a lemma $\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j$ remain the same. However, when we add the conclusions ψ_j of the lemma in the hypotheses of the ordered clause, we need to assign ordering functions on all facts of ψ_j. Since lemmas are satisfied on all traces T, we know that if $F_1\sigma, \dots, F_n\sigma$ are satisfied at steps $\tau_1, \dots, \tau_n$ respectively then the facts in $\psi_j\sigma$ are satisfied for some τ such that $\tau \leq \max(\tau_1, \dots, \tau_n)$. On an ordered clause, we cannot compute the max of step since we only have ordering functions $\delta_1, \dots, \delta_n$ of $\lceil F_1\sigma \rceil^{sure}, \dots, \lceil F_n\sigma \rceil^{sure}$ as information. The next best thing is to consider an ordering function that subsumes all functions $\delta_1, \dots, \delta_n$. Given a conjunction of atomic formulas ψ and an ordering function δ, we denote by $\lceil \psi \rceil_{\delta}^{sure}$ the formula obtained from $\lceil \psi \rceil^{sure}$ by replacing every fact F in $\lceil \psi \rceil^{sure}$ by the ordered fact F^{δ}.

$$\frac{\begin{array}{c}\mathbb{C} \cup \{H \to C\} \qquad (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j) \in \mathcal{L} \\ \text{for all } i, \quad pred(F_i) \in \mathcal{S}_p \quad F_i' = \lceil F_i\sigma \rceil^{sure} \quad F_i'^{\delta_i} \in H \quad \delta \sqsupseteq_o \delta_i\end{array}}{\mathbb{C} \cup \{H \wedge \lceil \psi_j\sigma \rceil_{\delta}^{sure} \to C\}_{j=1}^{m}} \, (\mathrm{Lem}_o(\mathcal{L}, \mathcal{S}_p))$$

Application of inductive lemmas Similarly to standard lemmas, we will need to add an ordering function on the application of an inductive lemma. Moreover, we also need to modify the application conditions of the rule. First, note that the clauses (Rl) and the projection clauses ($\mathrm{Rf}_{\pi_i^g}$) for data constructor function symbols are not in $\mathbb{C}_{sat}$, which simplifies the application conditions. Second, in rule $\mathrm{Ind}(\mathcal{L}, \mathcal{S}_p)$ for the saturation procedure, we do not need to verify that the hypotheses on which we apply the inductive lemma satisfy the order $<_{ind}$ since the derivation we consider satisfies the trace w.r.t. $\vdash_{IO}^{\kappa_{io}}$, i.e. we know that the facts occur strictly before the conclusion and so they satisfy the order $<_{ind}$. In the verification procedure, we do not have this property since the conclusion of an ordered clause contains possibly multiple facts. However, thanks to the ordering functions, we can still determine when to apply the inductive lemma.

Definition 29. *Let $n \in \mathbb{N}$. Let $\delta_1, \dots, \delta_m$ ordering functions with $\mathrm{dom}(\delta_i) \subseteq \{1, \dots, n\}$ for all $i = 1 \dots m$. We say that $\delta_1, \dots, \delta_m$* are *$n$-*strict *when :*

1. *either for all $k \in \{1, \dots, m\}$, there exists $i \in \{1, \dots, n\}$ such that $\delta_k(i) = <$*
2. *or $m \leq n$ and there exists $\{j_1, \dots, j_m\} \subseteq \{1, \dots, n\}$ such that:*
   (a) *for all $k \in \{1, \dots, m\}$, $\delta_k(j_k)$ is defined;*
   (b) *for all $k, k' \in \{1, \dots, m\}$, $k \neq k'$ implies $j_k \neq j_{k'}$;*
   (c) *if $n = m$ then there exists $k \in \{1, \dots, m\}$ such that $\delta_k(j_k) = <$.*

Intuitively, when $\delta_1, \dots, \delta_m$ are n-strict, we can apply our inductive lemmas on the facts annotated by $\delta_1, \dots, \delta_m$. This is demonstrated in the following lemma.

Lemma 14. *Let $n \in \mathbb{N}$. Let $\delta_1, \dots, \delta_m$ be n-strict ordering functions. Let $\tau_1, \dots, \tau_n$ and $\tau_1', \dots, \tau_m'$ steps. If for all $i \in \{1, \dots, n\}$, for all $j \in \{1, \dots, m\}$, $\delta_j(i)$ defined implies τ_j' $\delta_j(i)$ τ_i then $\{\!\{\tau_1', \dots, \tau_m'\}\!\} <_m \{\!\{\tau_1, \dots, \tau_n\}\!\}$.*

Proof. Let us do a case analysis on which item of Definition 29 is satisfied by $\delta_1, \dots, \delta_m$.

- Item 1: In such a case, we know that for all $j \in \{1,\ldots,m\}$, there exists $i \in \{1,\ldots,n\}$ such that $\delta_j(i) = <$ implying that $\delta_j(i)$ is defined. Thus by hypothesis, $\tau'_j < \tau_i \leq \max(\tau_1,\ldots,\tau_n)$. Since for all $j \in \{1,\ldots,m\}$, $\tau'_j < \max(\tau_1,\ldots,\tau_n)$, we directly obtain that $\{\!\{\tau'_1,\ldots,\tau'_m\}\!\} <_m \{\!\{\tau_1,\ldots,\tau_m\}\!\}$.
- Item 2: Thanks to Item 2b, we know that the indices $j_1,\ldots,j_m$ are distinct and by Item 2a that $\delta_k(j_k)$ is defined for all $k \in \{1,\ldots,m\}$. By being defined, we deduce that for all $k \in \{1,\ldots,m\}$, $\tau'_k\ \delta_k(j_k)\ \tau_{j_k}$ meaning that $\tau'_k \leq \tau_{j_k}$. Hence, we have show that the steps $\tau_1,\ldots,\tau_m$ are smaller than m distinct steps of the multiset $\{\!\{\tau_1,\ldots,\tau_n\}\!\}$. When $m < n$, it directly entails that $\{\!\{\tau'_1,\ldots,\tau'_m\}\!\} <_m \{\!\{\tau_1,\ldots,\tau_n\}\!\}$. When $m = n$, Item 2c guarantees that at least one of the ordering functions is strict, i.e. $\delta_k(j_k) = <$, and so $\tau'_k < \tau_{j_k}$. This allows us to conclude that $\{\!\{\tau'_1,\ldots,\tau'_m\}\!\} <_m \{\!\{\tau_1,\ldots,\tau_n\}\!\}$. □

We can now state our transformation rule for inductive lemmas.

$$\frac{\begin{array}{c}\mathbb{C} \cup \{F_1'^{\delta_1} \wedge \ldots \wedge F_m'^{\delta_m} \wedge H \rightarrow \curlywedge_{i=1}^n F_i''\} \qquad \delta_1,\ldots,\delta_m \text{ are } n\text{-strict} \\ (\bigwedge_{i=1}^m F_i \Rightarrow \bigvee_{j=1}^k \psi_j) \in \mathcal{L} \qquad \forall i, pred(F_i) \in \mathcal{S}_p,\ \lceil F_i\sigma \rceil^{sure} = F_i' \text{ and } \delta \sqsupseteq_o \delta_i\end{array}}{\mathbb{C} \cup \{F_1'^{\delta_1} \wedge \ldots \wedge F_m'^{\delta_m} \wedge H \wedge \lceil \psi_j\sigma \rceil_\delta^{sure} \rightarrow \curlywedge_{i=1}^n F_i''\}_{j=1}^k}\ (\mathrm{Ind}_o(\mathcal{L},\mathcal{S}_p))$$

6.4 The procedure and its soundness

The procedure is the same as the saturation procedure except that we rely on the transformation rules described in this section. We will denote $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$, $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C})$ and $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C},\mathbb{C}_{sat})$ the corresponding algorithms for verification of correspondence queries.

Theorem 4. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates containing the event predicates* m-event *and* s-event*. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant lemmas. Let $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}$. Let $\mathbb{C}$ be a set of well originated, simplified, selection free clauses containing the selection free clauses of $\mathbb{C}_{std}$.*

For all ordered clauses R satisfying $\mathcal{S}_p$, for all ordered derivations $\mathcal{D}$ of $\curlywedge_{i=1}^m F_i$ at steps $\tilde{\tau}$ from $\{R\}$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T,\tilde{\tau})$ and $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$, there exists an ordered derivation $\mathcal{D}'$ of $\curlywedge_{i=1}^m F_i$ at steps $\tilde{\tau}$ from $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\{R\},\mathbb{C})$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}'$.

The ordered clause R in the Theorem 4 is typically the initial clause we generate from the query we are proving. For example, when proving a query $\mathsf{event}(A) \wedge \mathsf{event}(B) \Rightarrow \mathsf{event}(C)$, we will apply the solving procedure with the ordered clause m-event$(x,A)^{\delta_A} \wedge$ m-event$(y,B)^{\delta_B} \rightarrow$ m-event$(x,A) \curlywedge$ m-event(y,B) where δ_A is defined only on 1 with $\delta_A(1) = \leq$ and δ_B is defined only on 2 with $\delta_B(2) = \leq$. This ordered clause typically requests all derivations that satisfy both $\mathsf{event}(A)$ and $\mathsf{event}(B)$. Note that we can use the ordered clause to verify more complex requests. For instance, by defining the same ordered clause but with a δ_B being defined as $\delta_B(1) = \delta_B(2) = \leq$, we are essentially requesting all derivations where both events A and B are satisfied *and* the event B must have occurred before event A.

7 The verification procedure

In this section we explain how we verify a query from the set of saturated of clauses. The complete procedure is quite complex as we need to be able to handle queries that may contain nested queries and injective events at the same time. For clarity sake, we will start with simpler queries which will allow us to explain the different components of the procedure separately and then we will show how to combine all of them to handle the most complex queries.

In Section 2.4, we made the distinction between axioms and lemmas, with lemmas being intermediate correspondence properties that ProVerif needs to prove before proving the main queries; and with axioms being similar to lemmas but do not need to be proved since they are typically guaranteed by other means (e.g. proof by hand). In this section however, as we explain the verification procedures for a set of queries $\mathcal{Q}$, we will consider a set of lemmas $\mathcal{L}$ guaranteed to hold, independently of how they have been proved. As such, in the statement of our algorithms and theorems, we will assimilate axioms and lemmas and only talk about a set of lemmas.

7.1 Equivalence queries

Equivalence queries are the simplest queries in term of verification as most of the work is done during the saturation procedure. In fact they are the only queries that do not rely on the solving procedure as we will only look at whether bad is derivable or not. Moreover, equivalence queries are not proved by induction meaning that the set $\mathcal{L}_i$ in the saturation procedure is empty. Finally, the set of predicates $\mathcal{S}_p$ used in the saturation and solving procedures matters to guarantee that facts occur strictly before other. The complete algorithm for verifying equivalence queries is given in Algorithm 1.

Algorithm 1: `prove'`$(\mathcal{C}_I, \mathcal{L}, \mathcal{R})$: Verification procedure for equivalence queries

input: An initial instrumented biconfiguration $\mathcal{C}_I$, sets of lemmas $\mathcal{L}$ and restrictions $\mathcal{R}$ on bitraces

$\kappa_{io} :=$ the smallest natural number such that all restrictions in $\mathcal{R}$ are fully IO-κ_{io}-compliant

$\mathbb{C} := \mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$

$\mathcal{S}_p$ is the set of all predicates in $\mathcal{L} \cup \mathcal{R}$

$\mathbb{C}_{sat} := \mathsf{saturateB}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\emptyset}(\mathbb{C})$

return $\neg\exists\ (H \to \text{bad}) \in \mathbb{C}_{sat}$

The soundness of the algorithm is given in the following theorem.

Theorem 5. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial biconfiguration and $\mathcal{C}_I$ be its associated initial instrumented biconfiguration. Let $\mathcal{L}$, $\mathcal{R}$ be respectively sets of lemmas and restrictions.*

If the following holds:

- *for all $\varrho \in \mathcal{L}$, $names(\varrho) \subseteq \mathcal{E}$ and $(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \to_{o'})_{|\mathcal{R}}) \models \varrho$*
- `prove'`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{R}]_i)$ *terminates and returns true*

then $\mathsf{trace}(\mathcal{C}, \to_{o'})_{|\mathcal{R}}\downarrow\uparrow$.

Proof. By Lemma 4, $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}\Downarrow\Uparrow$ is equivalent to $\mathsf{trace}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}\Downarrow\Uparrow_i$. By Lemma 9, we deduce that $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}\Downarrow\Uparrow$ is equivalent to $\mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}\Downarrow\Uparrow_i$. Similarly, relying on Lemmas 5 and 10, we know that for all $\varrho \in \mathcal{L}$, $(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}) \models \varrho$ implies $(\vdash^{\kappa}_{IO'}, \vdash_{i'}, \mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}) \models \varrho$.

Let us now assume by contradiction that $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}\Downarrow\Uparrow$ does not hold. Hence, $\mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}\Downarrow\Uparrow_i$ does not hold. This implies that there exists a bitrace $T \in \mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_{i'})_{|[\mathcal{R}]_i}$ such that $\neg T\Downarrow\Uparrow$. By lemma 11, there exists a derivation $\mathcal{D}$ of bad at step τ from $\mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}^{<\tau}_{e'}(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}$.

Denoting $\mathbb{C} = \mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$, we now apply Theorem 3 (by considering the empty set for the set of inductive lemmas, and the set $[\mathcal{L}]_i$ for the set of lemmas) which allow us to deduce that there exists a derivation $\mathcal{D}'$ of bad from $\mathsf{saturateB}^{\mathcal{S}_p}_{[\mathcal{L}]_i \cup [\mathcal{R}]_i, \emptyset}(\mathbb{C}) \cup \mathbb{C}^{<\tau}_{e'}(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash' \mathcal{D}$.

By construction, $\mathcal{D}'$ being a derivation of bad implies that the child of the root of D' must be labeled by a clause of the form $H \rightarrow$ bad. Since $\mathbb{C}^{<\tau}_{e'}(T)$ does not contain any clause concluding bad, we deduce that $(H \rightarrow \text{bad}) \in \mathsf{saturateB}^{\mathcal{S}_p}_{[\mathcal{L}]_i \cup [\mathcal{R}]_i, \emptyset}(\mathbb{C})$. However, by hypothesis, `prove'`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{R}]_i)$ terminates and returns *true*, which implies by definition that $\neg\exists\ (H \rightarrow \text{bad}) \in \mathsf{saturateB}^{\mathcal{S}_p}_{[\mathcal{L}]_i \cup [\mathcal{R}]_i, \emptyset}(\mathbb{C})$. Hence we obtain a contradiction and so $\mathsf{trace}(\mathcal{C}, \rightarrow_{o'})_{|\mathcal{R}}\Downarrow\Uparrow$ holds. □

7.2 Simple correspondence queries

A simple correspondence query does not contain nested queries nor injective events, thus the conclusion of the query may still contain attacker, message and table facts, events, and generic predicates. Events are the simplest to check since they are never reduced by the saturation or solving procedure. Hence, we only need to check whether they occur in the hypotheses of the clauses obtained through the solving procedure.

This is however not the case of attacker, message and table facts. For example, a fact $\text{att}_{\kappa}((a, b))$ would never occur in the hypotheses of the clauses since it would be split in $\text{att}_{\kappa}(a) \wedge \text{att}_{\kappa}(b)$ by the transformation rule DataHyp. For a query $\text{event}(A) \Rightarrow \text{att}_{\kappa}((a, b))$, it is not sufficient to check whether the fact $\text{att}_{\kappa}((a, b))$ occurs in the hypotheses of the clauses obtained through the solving procedure. To be more specific, it would be sound but would yield too many false positives as the procedure would always fail. Furthermore, the clause we generate ensures that the facts are satisfied w.r.t. $\vdash^{\kappa_{io}}_{IO}$ which is stronger than the satisfaction relation $\vdash_i$ for the attacker and table facts. Therefore, in the following definition, we show how to prove a fact w.r.t. $\vdash_i$ from the hypotheses of a clause.

Definition 30. *Let F be an attacker, message or table fact. Let $F_1, \ldots, F_n$ be a set of facts. We say that F is deducible from $F_1, \ldots, F_n$ under ϕ if there exists a partial derivation $\mathcal{D}$ of F from $\{(RInit), (RGen), (RFail), (Rf), (Rap), (Rtp)\}$ such that:*

- *all facts in $\mathbb{F}_{us}(\mathcal{D})$ are in $\{F_1, \ldots, F_n\}$*
- *ϕ is the conjunction of all formulae in $\mathbb{F}_{us}(\mathcal{D})$*

Intuitively, $F_1, \ldots, F_n$ represent the hypotheses of the clause we obtain through the solving procedure. Notice that we only allow in the partial derivation the rules (RInit), (RGen), (RFail), (Rf), (Rap) and (Rtp) which correspond to either initial knowledge of the intruder (rules (RInit) and (RFail)) or the application of the transition rule I-App, I-New, I-Phase

(rules (Rf),(Rap) and (Rtp)). Note that by requiring that the facts in $\mathbb{F}_{us}(\mathcal{D})$ are in $\{F_1, \ldots, F_n\}$, we ensure that the variables of $\{F_1, \ldots, F_n\}$ are not instantiated and so the derivation would hold for any instantiations of the variables of $\{F_1, \ldots, F_n\}$.

The last difficulty comes from checking disequalities and inequalities. When the query requires to prove a disequalities, say $M \neq N$, ProVerif 2.00 directly checks whether the hypothesis of the clause and in particular its disequalities imply the ones of the query. However, when the conclusion of the query contains disjunctions, this implication of disequalities does not allow ProVerif to prove simple queries.

Example 22. Consider the query $\mathsf{event}(A(x)) \Rightarrow \mathsf{event}(B(a)) \vee x \neq a$ and assume the solving procedure generates the clause s-event$(o_1[], B(x)) \rightarrow$ m-event$(o_2[], A(x))$. In ProVerif 2.00, the verification would fail since the hypothesis of the clause does not imply $x \neq a$ and $\mathsf{event}(B(a))$ does not match $\mathsf{event}(B(x))$. In this paper, based on the algorithm presented in [CCT18], we try to prove the query on a *complete coverage* of the clause. In our example, the clause s-event$(o_1[], B(x)) \rightarrow$ m-event$(o_2[], A(x))$ would be split into s-event$(o_1[], B(x)) \wedge x = a \rightarrow$ m-event$(o_2[], A(x))$ and s-event$(o_1[], B(x)) \wedge x \neq a \rightarrow$ m-event$(o_2[], A(x))$. The latter would imply $x \neq a$ trivially, whereas the former would be simplified into s-event$(o_1[], B(a)) \rightarrow$ m-event$(o_2[], A(a))$ and so would also satisfy the query. ▶

We formally define the notion of complete coverage as follows.

Definition 31. *Let $H \rightarrow C$ be an ordered clause. Let $\mathbb{C}$ be a set of clauses. We say that $\mathbb{C}$* is a complete coverage of $H \rightarrow C$ *when $\mathbb{C} \equiv \{H \wedge \phi_i \rightarrow C\}_{i=1}^n$ where:*

- *for all $i \in \{1, \ldots, n\}$, $fv(\phi_i) \subseteq fv(H, C)$*
- $\phi_1 \vee \ldots \vee \phi_n \equiv \top$

Given two sets of clauses $\mathbb{C}, \mathbb{C}'$. We say that $\mathbb{C}$ completely covers $\mathbb{C}'$ when $\mathbb{C} = \bigcup_i^n \mathbb{C}_i$ (as set equality), $\mathbb{C}' = \{H_i \rightarrow C_i\}_{i=1}^n$ and for all $i \in \{1, \ldots, n\}$, $\mathbb{C}_i$ is a complete coverage of $H_i \rightarrow C_i$.

Example 23. Coming back to Example 22, the following set $\mathbb{C}$ is a complete coverage of s-event$(o_1[], B(x)) \rightarrow$ m-event$(o_2[], A(x))$:

$$\mathbb{C} = \left\{ \begin{array}{l} \text{s-event}(o_1[], B(x)) \wedge x = a \rightarrow \text{m-event}(o_2[], A(x)), \\ \text{s-event}(o_1[], B(x)) \wedge x \neq a \rightarrow \text{m-event}(o_2[], A(x)) \end{array} \right\}$$

▶

Queries extended with event occurrences The verification of a simple correspondence query then tries to find one disjunct in the conclusion of the query that matches the hypothesis of the ordered clauses generated by the solving procedure. Recall that in a correspondence query on traces, we use the predicates event and inj_k-event with arity one (for the event) in both premises and conclusions of queries. However, in ordered clauses, we rely on two different predicates: (i) s-event(o, ev) for events in conclusion (ii) m-event(o, ev) for events in hypotheses In particular, note that the event predicates in ordered clauses contain the occurrence name of the event giving more information on when this event was executed. Querying the event event$(A(M))$ can be seen as showing that $A(M)$ was executed without any condition on its occurrence. However, for nested queries and injective queries, it will

be useful to also query events with some conditions on their occurrence. Thus, from now on, we consider that queries rely on the overloaded predicates event and inj_k-event that take into account the occurrence (see Section 3 for the semantics). Note that query with standard event$(A(M)$ (resp. inj_k-event$(A(M))$) can be easily transformed into an equivalent query with the overloaded definitions of events by adding a fresh variable in the occurrence argument, i.e. event$(x, A(M)$ (resp. inj_k-event$(x, A(M))$) where x is fresh.

Finally, in Section 5, we already introduced the operator $\lceil\psi\rceil^{sure}$ that replaces in ψ all events event(ev) (resp. inj_k-event(ev)) with a sure-event s-event(x, ev) with x a fresh variable. We thus consider the same transformation for our new event predicates, that is $\lceil\text{event}(o, ev)\rceil^{sure} = \lceil\text{inj}_k\text{-event}(o, ev)\rceil^{sure} = \text{s-event}(o, ev)$. Similarly, we consider the operator $\lceil\phi\rceil^{may}$ that replace in ϕ the facts event(o, ev) and inj_k-event(o, ev) with may-event, i.e., $\lceil\text{event}(o, ev)\rceil^{may} = \lceil\text{inj}_k\text{-event}(o, ev)\rceil^{may} = \text{m-event}(o, ev)$. We also extend these definitions by replacing all bi-events event$'(ev, ev)$ by m-event$'(ev, ev')$ and s-event$'(ev, ev')$ respectively.

Checking the query. Recall that queries are in disjunctive normal form. Hence, a query ϱ is of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j$ where ψ_j is a conjunction of atomic formulae. Thus to check a query against an ordered clause R, we need to find a substitution instantiating the facts of one of the disjuncts that match the facts of the clause. In other words, we always check whether there exists $j \in \{1, \ldots, m\}$ such that R satisfies the subquery $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j$. Such a subquery is called a *disjunct query*. Formally, a disjunct query is a non-nested and disjunction free query.

Definition 32 (Solution of a disjunct query)**.** *Let $\varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow H \wedge \phi)$ be a disjunct query. A solution of ϱ is a pair (R, σ) where σ is a substitution and $R = (H' \wedge \phi' \rightarrow \bigwedge_{i=1}^{n} F'_i)$ is an ordered clause such that:*

- *for all $i \in \{1, \ldots, n\}$, $\lceil F_i\sigma\rceil^{may} = F'_i$*
- *for all (inj-)event $F \in H$, either there exists $F'^{\delta'} \in H'$ such that $\lceil F\sigma\rceil^{sure} = F'$ or there exists $i \in \{1, \ldots, n\}$ such that $\lceil F\sigma\rceil^{may} = F'_i$*
- *$\phi' \models \phi\sigma$*
- *for all attacker, message and table facts F in H, $F\sigma$ is deducible from $\{F \in H' \mid pred(F) \in \mathcal{S}_p\}$ under some ϕ'' such that $\phi' \models \phi''$.*

We denote $R, \sigma \models \varrho$ when (R, σ) is a solution of ϱ.

The first item of the definition requires that premises of the query match the conclusion of the clause after instantiation by σ. The second item states that the events in the query are satisfied when either they occur in the hypotheses of the clause or directly in its conclusion. Indeed, the semantics of the satisfiability of the query does not proscribe two events in the conclusion and premises of the query to be matched by the same event in the trace. For example, the query event$(A) \Rightarrow$ event(A) is always true.

Example 24. Let $\varrho = \text{event}(x, A(x')) \wedge \text{event}(y, B(y')) \Rightarrow \text{event}(z, C(x', y')) \wedge \text{att}_1((x', y')) \wedge x' \neq y'$. Assume that a name $a[]$ is in the initial knowledge of the attacker. We have $R, \sigma \models \varrho$ with:

- $R = \text{s-event}(o_3[], C(a[], y_1)) \wedge y_1 \neq a[] \wedge \text{att}_0(y_1) \rightarrow \text{m-event}(o_1[], A(a[])) \curlywedge \text{m-event}(o_2[], B(y_1))$

- $\sigma = \{x \mapsto o_1[]; x' \mapsto a[]; y \mapsto o_2[]; y' \mapsto y_1; z \mapsto o_3[]\}$

Since we assume that $a[]$ is in the initial knowledge of the attacker, we know that $\rightarrow \text{att}_0(a[])$ is amongst the rules (RInit). Using the rule allowing the intruder to change phases (Rap) and the application of tuple from (Rf), we can build a partial derivation $\mathcal{D}$ of $\text{att}_1((a[], y_1))$ where $\mathbb{F}_{us}(\mathcal{D}) = \text{att}_0(y_1)$. ▶

The main procedure. The complete verification procedure, defined in Algorithm 2, starts by generating the set of inductive lemmas $\mathcal{L}_i$ from the queries in $\mathcal{Q}$. It then generates the set of allowed predicates $\mathcal{S}_p$. Note that $\mathcal{S}_p$ does not gather the predicates that are in the premises of the queries but only in their conclusion. Indeed, as mentioned in Section 4, facts in the conclusion of the queries will be matched with hypotheses of Horn clauses. Hence, we need to ensure the relation order between the facts in the hypotheses of the clauses and their conclusion. Since transformation rules such as general redundancy preserves such properties only on predicates in $\mathcal{S}_p$, we need to include the predicates of the conclusion of the queries inside $\mathcal{S}_p$. The third and fourth steps of the algorithm consist of generating the initial clauses and saturating them, yielding the set of ordered saturated clause $\mathbb{C}_{sat}$. These four steps are in fact common to every verification procedure for correspondence queries (nested, injective, simple). The final step consists of verifying each query in $\mathcal{Q}$. Note that when we write **return** ϕ with ϕ being a formula containing some quantifiers, we consider that the function returns *true* if the formula is true, and *false* otherwise.

The verification function starts by solving the ordered clause, yielding the set $\mathbb{C}_s$. We distinguish whether the query ϱ is in $\mathcal{L}_i$ or not. Intuitively, when we try to prove the lemmas in $\mathcal{L}_i$, we solve the ordered clause by considering the lemmas in $\mathcal{L}_i$ as inductive hypothesis, i.e. we apply $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$. Once all lemmas in $\mathcal{L}_i$ are proved, we can prove the remaining queries by considering the lemmas in $\mathcal{L}_i$ similarly to the lemmas in $\mathcal{L}$, i.e. we apply $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i\cup\mathcal{R},\emptyset}(\{R_q\}, \mathbb{C}_{sat})$.

Theorem 4 indicates that for all derivations from the request clause, there exists a derivation of the same facts from the set $\mathbb{C}_s$. Thus, by verifying that all rules in $\mathbb{C}_s$ satisfy the query, we guarantee that the derivation corresponding to the concrete facts of the trace also satisfy the query. The function `verify` as described in Algorithm 2 is non-deterministic due to the existential checks. In the implementation, the pair $(\sigma, \mathbb{C}_s)$ is generated by continuation on the fly until the algorithm finds one pair that verifies all conditions.

7.3 Nested queries

When verifying that an ordered clause R and a substitution σ are solution of a disjunct query ϱ, we intuitively verify that events in the conclusion of ϱ occur before one of the facts in ϱ's premises. However, proving a disjunct containing a nested query requires to order different facts within the conclusion of the disjunct. This is not directly possible in our setup, even with ordering functions in ordered clauses as they order facts from hypotheses in a clause w.r.t. facts in its conclusion. Thus, to prove a nested query, we proceed in two steps as described in the following example.

Example 25. Consider a query $\text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2)) \rightsquigarrow \text{event}(z, C(M_3))$. We start as if we are proving the simple correspondence query $\text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2))$ hence

1. Saturating the set of saturated clauses $\mathbb{C}_{sat}$

Algorithm 2: Verification procedure for simple correspondence queries

input: An initial instrumented configuration $\mathcal{C}_I$, a set of instrumented lemmas $\mathcal{L}$ on traces, a set of instrumented queries $\mathcal{Q}$, a set of restrictions $\mathcal{R}$

Function `verify`(ϱ, R_q)

- **Data:** ϱ is a query of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j$ where ψ_js are conjunctions of facts and generic predicates
- **Data:** R_q is an ordered clause
- **if** $\varrho \in \mathcal{L}_i$ **then**
  - $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$
- **else**
  - $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i\cup\mathcal{R},\emptyset}(\{R_q\}, \mathbb{C}_{sat})$
- **return** $\exists \mathbb{C}'_s$ *completely covering* $\mathbb{C}_s$
  - $\forall R \in \mathbb{C}'_s$
    - $\exists j \in \{1, \ldots, m\}$ *and* σ
      - $R, \sigma \models \bigwedge_{i=1}^{n} F_i \Rightarrow \phi_j$

`/* Generation of the inductive lemmas and allowed predicates */`

$\mathcal{L}_i := \{\varrho^{ind} \mid \varrho \in \mathcal{Q} \wedge \varrho^{ind}$ does not have $\top$ as conclusion$\}$

$\mathcal{S}_1$ is the set of all predicates in the conclusions of queries in $\mathcal{Q}$

$\mathcal{S}_2$ is the set of all predicates in $\mathcal{L}, \mathcal{L}_i, \mathcal{R}$

$\mathcal{S}_p := \mathcal{S}_1 \cup \mathcal{S}_2 \cup \{\text{att}_i \mid \text{att}_j \in \mathcal{S}_1 \wedge i \leq j\} \cup \{\text{table}_i \mid \text{table}_j \in \mathcal{S}_1 \wedge i \leq j\} \cup \{\text{m-event}, \text{s-event}\}$

`/* Generation of initial clauses */`

$\kappa_{io} :=$ the smallest natural number such that all queries in $\mathcal{Q}$ are IO-κ_{io}-compliant and all restrictions in $\mathcal{R}$ are fully IO-κ_{io}-compliant

$\mathbb{C} := \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$

`/* Saturation */`

$\mathbb{C}_{sat} := \mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\mathbb{C})$

`/* Verification */`

return $\forall \varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \psi) \in \mathcal{Q} \cup \mathcal{L}_i$

- $G_1 := \lceil F_1 \rceil^{may}, \ldots, G_n = \lceil F_n \rceil^{may}$
- $R_q := (G_1^{[1\mapsto\leq]} \wedge \ldots \wedge G_n^{[n\mapsto\leq]} \rightarrow \curlywedge_{i=1}^{n} G_i)$ `/* The ordered clause */`
- `verify`(ϱ, R_q)

2. Generating the ordered clause $R_q = \text{m-event}(x, A(M_1))^{\delta} \rightarrow \mathit{conj}_{[\text{m-event}]}(x, A(M_1))$ with $\delta = [1 \mapsto \leq]$

3. Solving the ordered clause $\mathbb{C}_s = \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$

In the verification of simple correspondence query, we would check that for all ordered clauses $R \in \mathbb{C}_s$, $R, \sigma \models \text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2))$ for some substitution σ. In other words, we check that R is of the form $H \wedge \text{s-event}(y\sigma, B(M_2\sigma))^{\delta} \rightarrow \mathit{conj}_{[\text{m-event}]}(x\sigma, A(M_1\sigma))$.

To prove the nested query $\text{event}(B(M_2)) \rightsquigarrow \text{event}(C(M_3))$, we use the substitution σ and generate the following new request clause R'_q:

$$H \wedge \text{s-event}(y\sigma, B(M_2\sigma))^{\delta} \wedge \text{m-event}(y\sigma, B(M_2\sigma))^{\delta'} \rightarrow \text{m-event}(x\sigma, A(M_1)\sigma) \curlywedge \text{m-event}(y\sigma, B(M_2\sigma))$$

where $\delta' = \delta \circ [2 \mapsto \leq]$. By adding $\text{m-event}(y\sigma, B(M_2\sigma))^{\delta'}$ in the hypothesis of the clause, we request to find the derivations that trigger the event $B(M_2\sigma)$ at occurrence $y\sigma$. Moreover, by also adding $\text{m-event}(y\sigma, B(M_2\sigma))$ in the conclusion of the request clause, we will be able to order the events of the derivation with respect to $\text{m-event}(y\sigma, B(M_2\sigma))$.

The procedure then solves R'_q resulting in a set $\mathbb{C}'_s$ and checks the ordered clauses in $\mathbb{C}'_s$ against the query $\text{event}(x\sigma, A(M_1\sigma)) \wedge \text{event}(y\sigma, B(M_2\sigma)) \Rightarrow \text{event}(z\sigma, C(M_3\sigma))^{[2 \mapsto \leq]}$ in which the ordered fact $\text{event}(z\sigma, C(M_3\sigma))^{[2 \mapsto \leq]}$ requires $\text{event}(z\sigma, C(M_3\sigma))$ *should occur at the same time or before* $\text{event}(y\sigma, B(M_2\sigma))$. This additional condition allows us to ensure that the nested query holds. ▶

In Example 25, we showed queries should allow conditions on the order between a fact of the hypotheses and a fact of the conclusion. Thus, we extend the syntax of the queries by allowing ordered facts. The satisfaction relation $\models$ for an annotated formula is also extended as follows: for all trace T, for all substitution σ and all tuples of steps $\tilde{\tau}$,

$$(\vdash_i, T, (\tilde{\tau}, \sigma)) \models F^{\delta,\mu} \quad \text{iff} \quad \mu(\tilde{\tau}, \sigma) \text{ is defined, } T, \mu(\tilde{\tau}, \sigma) \vdash_i F \text{ and } \forall i \in \mathit{Dom}\delta,\ \mu(\tilde{\tau}, \sigma)\ \delta(i)\ \tau_i$$

We will call *ordered queries* the queries containing ordered facts.

We also extend the definition of disjunction query to *ordered disjunct query* and their solutions (updating Definition 32) to take into account the ordering functions in the query. To ease the reading, we highlight in red the modifications w.r.t. Definition 32.

Definition 33 (Solution of an ordered disjunct query, Subsume Definition 32)**.** *Let* $\varrho = \bigwedge_{i=1}^{n} F_i \Rightarrow H \wedge \phi$ *be an ordered disjunct query. A solution of* ϱ *is a triple* $(R, \sigma, \mathbb{M})$ *where* σ *is a substitution,* $R = (H' \wedge \phi' \rightarrow \curlywedge_{i=1}^{n} F'_i)$ *is an ordered clause, and* $\mathbb{M}$ *is a multiset of pairs of ordered facts and ordering functions such that:*

- $\{\!\{F^{\delta} \in H \mid F \textit{ is an (inj-)event}\}\!\} = \{\!\{F^{\delta} \mid (F^{\delta}, \delta') \in \mathbb{M}\}\!\}$
- *for all* $i \in \{1, \ldots, n\}$, $\lceil F_i\sigma \rceil^{may} = F'_i$
- *for all* $(F^{\delta}, \delta') \in \mathbb{M}$, $\delta \sqsupseteq_o \delta'$ *and either there exists a fact* F' *such that* $F'^{\delta'} \in H'$ *and* $\lceil F\sigma \rceil^{sure} = F'$ *or there exists* $i \in \{1, \ldots, n\}$ *such that* $\lceil F\sigma \rceil^{may} = F'_i$ *and* $\delta' = [i \mapsto \leq]$
- $\phi' \models \phi\sigma$
- *for all attacker, message and table facts* F^{δ} *in* H, $F\sigma$ *is deducible from* $\{F' \mid F'^{\delta'} \in H' \wedge \mathit{pred}(F') \in \mathcal{S}_p \wedge \delta \sqsupseteq_o \delta'\}$ *under some* ϕ'' *such that* $\phi' \models \phi''$.

We denote $R, \sigma, \mathbb{M} \models \varrho$ when $(R, \sigma, \mathbb{M})$ is a solution of ϱ.

A solution $(R, \sigma, \mathbb{M})$ of an ordered disjunct query ϱ indicates that all events in the conclusion of the query ϱ can be found in the hypotheses of the rule R after instantiation by σ. However, since the query and the rule both contain ordering functions, we need to guarantee that the ordering functions in R imply the ones in the query ($\delta \sqsupseteq_o \delta'$ in the definition). We record in the multiset $\mathbb{M}$ the ordering function δ' in R with whom the ordered event F^{δ} from the query was matched. For attacker, message and table facts, similarly to Definition 32, we do not request that the fact after instantiation by σ directly occurs in the hypotheses but that it is deducible from the hypotheses of R.

The main procedure The procedure follows the same initial steps as for simple correspondence queries (Algorithm 2), i.e. generation of inductive lemmas, generation of set of allowed predicates, generation and saturation of initial clauses. The procedure differs in the function `verify` where nested queries has to be taken into account. The updated function `verify` is displayed in Algorithm 3 and the parts related to nested queries are highlighted in red.

The beginning of the function is the same, i.e. the ordered clause R_q is solved yielding the set of ordered clauses $\mathbb{C}_s$. Then the procedure finds a set $\mathbb{C}'_s$ completely covering $\mathbb{C}_s$ such that all ordered clauses $R = (H \to C)$ must be solution of at least one disjunct of the query (represented by j) with some substitution σ and some matching $\mathbb{M}$. Since the notion of ordered disjunct query does not include nested queries, we only keep the premises of the nested queries inside the disjunct when searching for solutions: $R, \sigma, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}}$. Note that when there is no nested query, i.e. $\ell_j = 0$, the disjunct query $R, \sigma, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j$ exactly corresponds to the one considered in simple correspondence query (the multiset $\mathbb{M}$ being ignored). When there are nested queries, i.e. $\ell_j \geq 1$, the procedure generates a new ordered clause R'_q and a new ordered query ϱ' that include the nested queries $\psi_{k,j}$ and recursively calls itself on (ϱ', R'_q).

The generation of the new ordered clause R'_q and query ϱ', described in the function `gen_nested`, follows the ideas given in Example 25. More specifically, the premises G_k of the nested queries $G_k^{\delta_k} \rightsquigarrow \psi_k$ are added in the conclusion and hypotheses of the ordered clause. In the hypotheses of the ordered clause, the fact G_k is associated to the ordering function $\delta_k[n+k \mapsto \leq]$. The part $[n+k \mapsto \leq]$ intuitively indicates that G_k *should occur before itself.* This is obviously always true but it allows to keep track of the facts deriving G_k when solving the ordered clause R'_q.

Finally, to build the ordered query ϱ', we replace the nested relations $G_k^{\delta_k} \rightsquigarrow \psi_k$ with $\psi_k^{\delta_k[n+k \mapsto \leq]}\sigma$ but we add $G_k\sigma$ in the premise of the query. The subquery $\psi_k^{\delta_k[n+k \mapsto \leq]}$ indicates that every fact in ψ_k has the ordering function $\delta_k[n+k \mapsto \leq]$. Once again, this intuitively means that *the facts in ψ_k should occur before the $n+k$-th facts of the conclusion*, i.e. $G_k\sigma$.

7.4 Injective queries

Checking simple correspondence queries or nested queries requires to show that each ordered clause generated by the solving procedure is a solution of the query for some substitution. Note that proving that ordered clauses are solutions of the query is done separately, i.e. ordered clauses are independent of each other. This is reflected by the first item of Definition 10.

Algorithm 3: Updated function `verify` for nested correspondence queries

Function `gen_nested`$(\varrho, (H \to C, \sigma, \mathbb{M}))$

Data: ϱ is an ordered query of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi \wedge \bigwedge_{k=1}^{\ell} G_k^{\delta_k} \rightsquigarrow \psi_k)$ where ψ_js are conjunctions of ordered facts and generic predicates

Data: $(H \to C, \sigma, \mathbb{M})$ is a solution of $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi \wedge \bigwedge_{k=1}^{\ell} G_k^{\delta_k}$

`/* Generate the nested request clause and corresponding query */`

Take $(G_1^{\delta_1}, \delta_1'), \ldots, (G_\ell^{\delta_\ell}, \delta_\ell') \in \mathbb{M}$

$R_q' := H \wedge \bigwedge_{k=1}^{\ell} \lceil G_k^{\delta_k'[n+k \mapsto \leq]} \rceil^{may} \sigma \to C \curlywedge \curlywedge_{k=1}^{\ell} \lceil G_k \rceil^{may} \sigma$

$\varrho' := \bigwedge_{i=1}^{n} F_i \sigma \wedge \bigwedge_{k=1}^{\ell} G_k \sigma \Rightarrow \psi\sigma \wedge \bigwedge_{k=1}^{\ell} \psi_k^{\delta_k[n+k \mapsto \leq]} \sigma$

return R_q', ϱ'

Function `verify`(ϱ, R_q)

Data: ϱ is an ordered query of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} (\psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j})$ where ψ_js are conjunctions of ordered facts and generic predicates

Data: R_q is an ordered clause

if $\varrho \in \mathcal{L}_i$ **then**

  $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L} \cup \mathcal{R}, \mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$

else

  $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L} \cup \mathcal{L}_i \cup \mathcal{R}, \emptyset}(\{R_q\}, \mathbb{C}_{sat})$

return $\exists \mathbb{C}_s'$ *completely covering* $\mathbb{C}_s$

  $\forall R = (H \to C) \in \mathbb{C}_s'$

    $\exists j \in \{1, \ldots, m\}$, *a substitution* σ, *and a matching* $\mathbb{M}$

      $R, \sigma, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}}$ **and**

      if $\ell_j \geq 1$ **then** `/* Nested queries */`

        $R_q', \varrho' =$ `gen_nested`$(\bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j}, (R, \sigma, \mathbb{M}))$

        `verify`(ϱ', R_q')

However for injective queries, we additionally need to verify a global injective property on all the solutions (item 2 and 3 of Definition 10).

Example 26. Consider the query $\text{inj}_1\text{-event}(x, A(M_1)) \Rightarrow \text{inj}_2\text{-event}(y, B(M_2))$. We start as if we are proving the simple correspondence query $\text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2))$, as in Example 25:

1. Saturating the set of saturated clauses $\mathbb{C}_{sat}$
2. Generating the annotated clause $R_q = \text{m-event}(x, A(M_1))^{\delta} \to conj_{[\text{m-event}]}(x, A(M_1))$ with $\delta = [1 \mapsto \leq]$
3. Solving the annotated clause $\mathbb{C}_s = \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L} \cup \mathcal{R}, \mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$

Once again, we check that all rules R in $\mathbb{C}_s$ are of the form $H \wedge \text{s-event}(y\sigma, B(M_2\sigma))^{\delta} \to$

$conj_{[\text{m-event}]}(x\sigma, A(M_1\sigma))$ for some σ, $\mathbb{M}$ and δ, i.e. $R, \sigma, \mathbb{M} \models \text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2))$.

For the purpose of this example, let us assume the $\mathbb{C}_s$ only contains two ordered clauses R_1 and R_2, as follows:

- $R_1 = H_1 \wedge \text{s-event}(y\sigma_1, B(M_2\sigma_1))^{\delta_1} \rightarrow conj_{[\text{m-event}]}(x\sigma_1, A(M_1\sigma_1))$
- $R_2 = H_2 \wedge \text{s-event}(y\sigma_2, B(M_1\sigma_2))^{\delta_2} \rightarrow conj_{[\text{m-event}]}(x\sigma_2, A(M_1\sigma_2))$

Now consider a concrete trace T and two steps τ_1, τ_2 such that $T, \tau_i \vdash_i \text{event}(x\sigma_i, A(M_1\sigma_i)$ for $i = 1, 2$. Theorems 2 and 4 indicate the events $\text{event}(y\sigma_1, B(M_2\sigma_1))$ and $\text{event}(y\sigma_2, B(M_2\sigma_2))$ are respectively executed before $\text{event}(x\sigma_1, A(M_1\sigma_1))$ and $\text{event}(x\sigma_2, A(M_1\sigma_2))$, i.e. there exists τ_1' and τ_2' such that $T, \tau_i' \vdash_i \text{s-event}(y\sigma_i, B(M_2\sigma_i))$.

Therefore, when checking the query, we can consider the partial function $\mu = [(\tau_1, \sigma_1) \mapsto \tau_1'; (\tau_2, \sigma_2) \mapsto \tau_2']$ for building the annotated query conclusion $\Psi = \text{inj}_2\text{-event}(y, B(M_2))^{\mu}$. However, in order to satisfy the injectivity property of the query (items 2 and 3 of Definition 2), we need to ensure that if $\tau_1 \neq \tau_2$ then $\tau_1' \neq \tau_2'$, i.e. if the events $\text{event}(x\sigma_1, A(M_1\sigma_1))$ and $\text{event}(x\sigma_2, A(M_1\sigma_2))$ do not occur at the same time then the events $\text{event}(y\sigma_1, B(M_2\sigma_1))$ and $\text{event}(y\sigma_2, B(M_1\sigma_2))$ do not occur at the same time in the trace T.

Ordered clauses do not directly give precise indications on the steps on which the events are satisfied. However, from Lemma 2, we can derive that the occurrence terms o in an event $\text{event}(o, ev)$ are unique in a trace, i.e. if two events with the same occurrence term are executed in the trace then these two events are the same events and are necessarily executed at the same time. Using this property, we can prove injectivity of the query by building the following clause and by checking that it is not satisfiable:

$$H_1 \wedge H_2 \wedge \text{s-event}(y\sigma_1, B(M_2\sigma_1)) \wedge \text{s-event}(y\sigma_2, B(M_1\sigma_2)) \wedge y\sigma_1 = y\sigma_2 \wedge x\sigma_1 \neq x\sigma_2$$
$$\rightarrow$$
$$\text{m-event}(x\sigma_1, A(M_1\sigma_1) \curlywedge \text{m-event}(x\sigma_2, A(M_2, \sigma_2))$$

In ProVerif 2.00, the injectivity check is also done by gathering the *environment* of events (a notion similar to our occurrence terms) and by then checking that the formula $y\sigma_1 = y\sigma_2 \wedge x\sigma_1 \neq x\sigma_2$ is unsatisfiable. In this work, we build the above clause and use the solving procedure to check whether the clause is satisfiable. This is strictly stronger as it allows us to apply ProVerif lemmas during the solving procedure which could remove false attacks. ▶

As illustrated in Example 26, the verification procedure needs to gather for each ordered clauses in the set of solved clauses the substitution σ that satisfy the query. In the implementation, the set of solutions is built by continuation while verifying the non-injective query. However, for sake of clarity, we present here the algorithm as if it first guesses the set of solutions and second verifies that it is indeed a set of solutions. In Algorithm 4, we present the updated function `verify` that takes as input the additional set $\mathcal{S}_{ol}$ and verifies that $\mathcal{S}_{ol}$ is a set of solutions. Note that $\mathcal{S}_{ol}$ is a set of tuples (R', σ', ϱ') where R' is an ordered clause, σ' is a substitution, and ϱ' is an ordered disjunct query. In Example 26, we ignored ϱ' as the query $\text{event}(x, A(M_1)) \Rightarrow \text{event}(y, B(M_2))$ does not contain any disjunction nor nested query. However, in the general case, we need to know to which disjunct ϱ' the 3-tuple $(R', \sigma', \mathbb{M})$ is solution of for some matching $\mathbb{M}$. As for the nested queries, we highlighted in Algorithm 4 the parts that were changed for injective queries.

Algorithm 4: Updated function `verify` for injective correspondence queries.

Function `verify`$(\varrho, R_q, \mathcal{S}_{ol})$

Data: ϱ is an ordered query of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m}(\psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j})$ where ψ_js are conjunctions of ordered facts and generic predicates

Data: R_q is an ordered clause

Data: $\mathcal{S}_{ol}$ is a set of tuples (R', σ', ϱ') where R' is an ordered clause, σ' is a substitution, ϱ' is an ordered disjunct query.

if $\varrho \in \mathcal{L}_i$ **then**

  $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\{R_q\}, \mathbb{C}_{sat})$

else

  $\mathbb{C}_s := \mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i\cup\mathcal{R},\emptyset}(\{R_q\}, \mathbb{C}_{sat})$

return $\exists \mathbb{C}'_s$ *completely covering* $\mathbb{C}_s$

  $\forall R = (H \rightarrow C) \in \mathbb{C}'_s$

    $\exists j \in \{1, \ldots, m\}$, *a substitution* σ, *and a matching* $\mathbb{M}$

      $R, \sigma, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}}$ **and**

      if $\ell_j \geq 1$ **then** `/* Nested queries */`

        $R'_q, \varrho' :=$ `gen_nested`$(\bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j}, (R, \sigma, \mathbb{M}))$

        `verify`$(\varrho', R'_q, \mathcal{S}_{ol})$

      else

        $(R, \sigma, \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_j) \in \mathcal{S}_{ol}$

Algorithm 5 describes the function `verify_inj` that checks whether a set of solutions $\mathcal{S}_{ol}$ verifies the injective property of the query, as illustrated in Example 26. Note that the function `verify_inj` also takes an integer n which was not mentioned in Example 26. Intuitively, when the query contains a nested query, the different disjuncts in the set of solution $\mathcal{S}_{ol}$ do not necessarily have the same number of facts in their premises since the function `gen_nested` generates new request clauses by adding facts in the premise (see Algorithm 3). However when verifying the injective property, we should only consider the premise of the initial query. Note that `gen_nested` only adds facts in the premise *after* (as a sequence) the initial facts, hence if n is the number of facts in the premise of the initial query then we know that the first n facts of a disjunct query's premise from $\mathcal{S}_{ol}$ are the ones corresponding to the initial query's premise. In Algorithm 5, we also rely on the function $\mathsf{occ}_n(\psi)$ to build the disequality in the clause. This function takes an integer n and a conjunction of facts as arguments and returns the tuple formed of occurrences of all injective events from the first n facts in the conjunction. Formally, $\mathsf{occ}_n(\bigwedge_{i=1}^{m} F_i) = (o_{i_1}, \ldots, o_{i_r})$ where $i_1, \ldots, i_r$ is the longest sequence of increasing indices such that for all $\ell \in \{1, \ldots, r\}$, $i_\ell \leq n$ and $F_{i_\ell} = \mathrm{inj}_{k_\ell}\text{-event}(o_{i_\ell}, ev_\ell)$ for some k_ℓ, ev_ℓ.

Example 27. Consider the initial query $\varrho = \mathrm{inj}_1\text{-event}(x, A(M_1)) \wedge \mathrm{att}_0(N) \Rightarrow \mathrm{inj}_2\text{-event}(y, B(M_2)) \rightsquigarrow \mathrm{inj}_3\text{-event}(z, C(M_3))$. Before verifying the nested query, the function `verify` will check the query $\varrho_1 = \mathrm{inj}_1\text{-event}(x, A(M_1)) \wedge \mathrm{att}_0(N) \Rightarrow \mathrm{inj}_2\text{-event}(y, B(M_2))$ hence yielding solutions of the form $(\sigma, H \rightarrow \text{m-event}(x, A(M_1)) \curlywedge \mathrm{att}_0(N), \varrho')$. The initial query only containing two facts in its premises, we apply $\mathsf{occ}_n(\cdot)$ with $n = 2$. Thus $\mathsf{occ}_2(\mathrm{inj}_1\text{-event}(x, A(M_1)) \wedge$

Algorithm 5: Function `verify_inj` checking injectivity of solutions.

Function `verify_inj`$(\mathcal{S}_{ol}, n)$

Data: $\mathcal{S}_{ol}$ is a set of tuples (R', σ', ϱ') where R' is an ordered clause, σ' is a substitution, ϱ' is a disjunct query.

Data: n represents the number of facts in the premise of the initial query

return $\forall (H_1 \to C_1, \sigma_1, \bigwedge_{i=1}^{n_1} F_{i,1} \Rightarrow \psi_1), (H_2 \to C_2, \sigma_2, \bigwedge_{i=1}^{n_2} F_{i,2} \Rightarrow \psi_2) \in \mathcal{S}_{ol}$

`/*` $H_1 \to C_1, H_2 \to C_2, \sigma_1, \sigma_2, \ldots$ `are assumed to have been freshly renamed` `*/`

$\forall \text{inj}_{k_1}\text{-event}(o_1, ev_1) \in \psi_1 \cup \{F_{i,1}\}_{i=n+1}^{n_1}$,

$\forall \text{inj}_{k_2}\text{-event}(o_2, ev_2) \in \psi_2 \cup \{F_{i,2}\}_{i=n+1}^{n_2}$

`/* tests item 2 of Definition 4 */`

if $k_1 = k_2$ **then**

$o_1' := \mathsf{occ}_n(\bigwedge_{i=1}^{n_1} F_{i,1})$

$o_2' := \mathsf{occ}_n(\bigwedge_{i=1}^{n_2} F_{i,2})$

$R_q := (H_1\sigma_1 \wedge H_2\sigma_2 \wedge o_1\sigma_1 = o_2\sigma_2 \wedge o_1'\sigma_1 \neq o_1'\sigma_2 \to C_1 \curlywedge C_2)$

$\mathsf{saturate}\mathsf{S}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i\cup\mathcal{R},\emptyset}(\{R_q\}, \mathbb{C}_{sat}) = \emptyset$

and

$\forall \text{inj}_{k_1}\text{-event}(o_1, ev_1) \in \psi_1 \cup \{F_{i,1}\}_{i=n+1}^{n_1}$,

$\forall \text{inj}_{k_2}\text{-event}(o_2, ev_2) \in \psi_1 \cup \{F_{i,1}\}_{i=n+1}^{n_1}$

`/* both events are from the same query` $\bigwedge_{i=1}^{n_1} F_{i,1} \Rightarrow \psi_1$ `*/`

`/* tests item 3 of Definition 4 */`

if $k_1 = k_2$ **then**

$(o_1\sigma_1, ev_1\sigma_1) = (o_2\sigma_1, ev_2\sigma_1)$

$\text{att}_0(N)) = x$. However, when checking the nested query, the function `verify` will call `gen_nested(`ϱ, R`)` with R an ordered clause, yielding the query $\varrho_2 = \text{inj}_1\text{-event}(x, A(M_1))\sigma' \wedge \text{att}_0(N)\sigma' \wedge \text{inj}_2\text{-event}(y, B(M_2))\sigma' \Rightarrow \text{inj}_3\text{-event}(z, C(M_3))^{\delta[3\mapsto\leq]}\sigma'$ for some σ', δ. Note that in ϱ_2, even though we have the presence of the injective event $\text{inj}_2\text{-event}(y, B(M_2))\sigma'$, the injectivity of the query should only be checked w.r.t. $\text{inj}_1\text{-event}(x, A(M_1))\sigma'$. This is reflected when applying $\mathsf{occ}_n(\cdot)$ with $n = 2$ since $\mathsf{occ}_2(\text{inj}_1\text{-event}(x, A(M_1))\sigma' \wedge \text{att}_0(N)\sigma' \wedge \text{inj}_2\text{-event}(y, B(M_2))\sigma') = x\sigma'$ which does not include the occurrence $y\sigma'$. ▶

Algorithm 6 presents the complete procedure for verifying correspondence queries. In particular, it subsumes Algorithm 2. As previously mentioned, the algorithm is presented as a non-deterministic procedure that guesses the set of solution $\mathcal{S}_{ol}$ and verifies both the non injective aspect of the query (`verify(`$\varrho, R_q, \mathcal{S}_{ol}$`)`) and the injective property of the query (`verify_inj(`$\mathcal{S}_{ol}, n$`)`).

The soundness of the algorithm is given in the following theorem.

Theorem 6. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial configuration and $\mathcal{C}_I$ be is associated initial instrumented configuration. Let $\mathcal{L}$ be a sets of lemmas. Let $\mathcal{R}$ be a set of restrictions. Let $\mathcal{Q}$ be a set of correspondence queries.*

If the following holds:

Algorithm 6: `prove`$(\mathcal{C}_I, \mathcal{L}, \mathcal{R}, \mathcal{Q})$: Verification procedure for injective nested correspondence queries

input: An initial instrumented configuration $\mathcal{C}_I$, a set of instrumented lemmas $\mathcal{L}$ on traces, a set of instrumented queries $\mathcal{Q}$

`/* Generation of the inductive lemmas and allowed predicates */`

$\mathcal{L}_i := \{\varrho^{ind} \mid \varrho \in \mathcal{Q} \wedge \varrho^{ind}$ does not have $\top$ as conclusion$\}$

$\mathcal{S}_1$ is the set of all predicates in the conclusions of queries in $\mathcal{Q}$

$\mathcal{S}_2$ is the set of all predicates in $\mathcal{L}, \mathcal{L}_i, \mathcal{R}$

$\mathcal{S}_p := \mathcal{S}_1 \cup \mathcal{S}_2 \cup \{\text{att}_i \mid \text{att}_j \in \mathcal{S}_1 \wedge i \leq j\} \cup \{\text{table}_i \mid \text{table}_j \in \mathcal{S}_1 \wedge i \leq j\} \cup \{\text{m-event}, \text{s-event}\}$

`/* Generation of initial clauses */`

$\kappa_{io} :=$ the smallest natural number such that all queries in $\mathcal{Q}$ are IO-κ_{io}-compliant and all restrictions in ∇ are fully IO-κ_{io}-compliant

$\mathbb{C} := \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$

`/* Saturation */`

$\mathbb{C}_{sat} := \mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\mathbb{C})$

`/* Verification */`

return $\forall \varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \psi_j) \in \mathcal{Q} \cup \mathcal{L}_i$

  $\exists \mathcal{S}_{ol}$ `/* Guess of a set of solutions */`

    $G_1 := \lceil F_1 \rceil^{may}, \ldots, G_n = \lceil F_n \rceil^{may}$

    $R_q := (G_1^{[1\mapsto\leq]} \wedge \ldots \wedge G_n^{[n\mapsto\leq]} \rightarrow \curlywedge_{i=1}^{n} G_i)$ `/* The request clause */`

    `verify`$(\varrho, R_q, \mathcal{S}_{ol})$ **and** `verify_inj`$(\mathcal{S}_{ol}, n)$

- *for all* $\varrho \in \mathcal{L} \cup \mathcal{Q} \cup \mathcal{R}$, $names(\varrho) \subseteq \mathcal{E}$
- *for all* $\varrho \in \mathcal{L}$, $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o))_{|\mathcal{R}} \models \varrho$
- `prove`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{R}]_i, [\mathcal{Q}]_i)$ *terminates and returns true*

then for all $\varrho \in \mathcal{Q}$, $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)_{|\mathcal{R}}) \models \varrho$.

7.5 Correspondence lemmas on bitraces

When proving a set $\mathcal{Q}$ of queries on bitraces, we rely in fact on the same procedure `prove`$(\mathcal{C}_I, \mathcal{L}, \mathcal{Q})$ in which we only modify the set of initial clauses $\mathbb{C}$ and the set of saturated clauses $\mathbb{C}_{sat}$. Following Theorem 3, the set of initial clauses $\mathbb{C}$ is such that $\mathbb{C}^c(\mathcal{C}_I, n_{IO}) \subseteq \mathbb{C} \subseteq \mathbb{C}'_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}'_{\mathcal{A}}(\mathcal{C}_I)$ where κ_{io} is the smallest natural number such that all queries in $\mathcal{Q}$ are IO-κ_{io}-compliant. We denote by `proveB`$(\mathcal{C}_I, \mathcal{L}, \mathcal{Q})$ the procedure for verifying the set of queries on bitraces $\mathcal{Q}$ in the initial biconfiguration $\mathcal{C}_I$.

The soundness of the algorithm is given in the following theorem.

Theorem 7. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial biconfiguration and $\mathcal{C}_I$ be is associated initial instrumented biconfiguration. Let $\mathcal{L}$ be a sets of* ProVerif *lemmas. Let $\mathcal{Q}$ be a set of correspondence queries on bitraces.*

If the following holds:

- *for all $\varrho \in \mathcal{L} \cup \mathcal{Q}$, $names(\varrho) \subseteq \mathcal{E}$*
- *for all $\varrho \in \mathcal{L}$, $(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \rightarrow_{o'})) \models \varrho$*
- `proveB(`$\mathcal{C}_I, [\mathcal{L}]_{i'}, [\mathcal{Q}]_{i'}$`)` *terminates and returns true*

then for all $\varrho \in \mathcal{Q}$, $(\vdash_{o'}, \mathsf{trace}(\mathcal{C}, \rightarrow_{o'})) \models \varrho$.

References

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *J. Log. Algebr. Program.*, 75(1):3–51, 2008.

[BBK17] Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (S&P'17)*, pages 483–503, May 2017.

[Bla09] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.

[Bla14] Bruno Blanchet. Automatic verification of security protocols in the symbolic model: The verifier proverif. In *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, volume 8604 of *Lecture Notes in Computer Science*, pages 54–87. Springer, 2014. `https://proverif.inria.fr`.

[Bla16] Bruno Blanchet. Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016.

[Bla17] Bruno Blanchet. Symbolic and computational mechanized verification of the ARINC823 avionic protocols. In *30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 68–82, Santa Barbara, CA, USA, August 2017. IEEE.

[BSCS17] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. Proverif 1.97: Automatic cryptographic protocol verifier, user manual and tutorial, July 2017.

[CB13] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with proverif. In David Basin and John Mitchell, editors, *Proceedings of the 2nd International Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *Lecture Notes in Computer Science*, pages 226–246, Roma, Italy, March 2013. Springer Berlin Heidelberg.

[CCK12] Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In *Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12)*, volume 7211, pages 108–127. Springer, 2012.

[CCT18] Vincent Cheval, Véronique Cortier, and Mathieu Turuani. A little more conversation, a little less action, a lot more satisfaction: Global states in proverif. In *Proceedings of the 31st IEEE Computer Security Foundations Symposium (CSF'18)*, Oxford, UK, July 2018. IEEE Computer Society Press.

[CGG19] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. *Belenios: A Simple Private and Verifiable Electronic Voting System*. Springer, 2019.

[CGT18] Véronique Cortier, David Galindo, and Mathieu Turuani. A formal analysis of the neuchâtel e-voting protocol. In *3rd IEEE European Symposium on Security and Privacy (EuroSP'18)*, pages 430–442, London, UK, April 2018.

[CKR18] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. Deepsec: Deciding equivalence properties in security protocols - theory and practice. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P'18)*, pages 525–542, San Francisco, CA, USA, May 2018. IEEE Computer Society Press.

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.

[KBB17] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 435–450. IEEE, 2017.

[KNB19] Nadim Kobeissi, Georgio Nicolas, and Karthikeyan Bhargavan. Noise explorer: Fully automated modeling and verification for arbitrary noise protocols. In *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*, pages 356–370. IEEE, 2019.

[Per16] Trevor Perrin. The noise protocol framework, 2016.

[Pra77] V. R. Pratt. Two easy theories whose combination is hard. Technical report, 1977.

[SMCB12] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In Stephen Chong, editor, *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 78–94. IEEE, 2012.

Index

A Proof of Lemma 8

The completeness of our restrictions is easy to prove.

Lemma 15. *Let $\mathcal{C}_I = \rho, P, \mathcal{A}$ be an initial instrumented configuration. Let $\kappa \in \mathbb{N}$. Let ψ be an IO-κ-compliant correspondence query such that $names(\psi) \subseteq dom(\rho)$.*

We have $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)) \models \psi$ implies $(\vdash^n_{IO}, \vdash_i, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i)) \models \psi$.

Proof. Note that the relation $\vdash^{n_{IO}}_{IO}$ implies $\vdash_i$, i.e. for all traces T, for all steps τ, for all facts F, $T, \tau \vdash^{n_{IO}}_{IO} F$ implies $T, \tau \vdash_i F$. Moreover, we also have $\mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i) \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Hence, the result holds. □

Proving the soundness of our restrictions is much harder. To do so, we introduce an intermediate satisfaction relation on facts, denoted $\vdash_{is}$, and that is defined as follows: $T, \tau \vdash_{is} F$ when $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and

- if $F = \mathrm{att}_m(M')$ then $m = n$, $M' = M\rho$ and $M \in \mathcal{A}$ for some M
- if $F = \mathrm{table}_m(tbl(M'_1, \ldots, M'_m))$ then $m = n$, $M'_1 = M_1\rho$, $\ldots$, $M'_m = M_m\rho$ and $tbl(M_1, \ldots, M_m) \in \mathcal{T}$ for some $M_1, \ldots, M_m$
- $T, \tau \vdash_i F$ otherwise

The satisfaction relation $\vdash_{is}$ is an intermediate relation between $\vdash_i$ and $\vdash^{n_{IO}}_{IO}$ that will allow us to prove the soundness of our restrictions in several stages.

Given a trace T, let us denote $steps(T) = \{1, \ldots, \mathsf{max}_{step}(T)\}$. Given a query $\psi = (F_1 \wedge \ldots \wedge F_n \Rightarrow \phi)$ and a trace T, let us denote $\mathsf{Prem}(\psi, T) = \{(\tau_1, \ldots, \tau_n, \sigma) \mid \tau_1, \ldots, \tau_n \in steps(T)$ and σ is a ground substitution with $dom(\sigma) = vars(F_1, \ldots, F_n)\}$. Moreover, given a satisfaction relation $\vdash$, given $\mathcal{P} \subseteq \mathsf{Prem}(\psi, T)$ and given an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$, let us denote $(\vdash, T, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$ when:

1. for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathcal{P}$, if $T, \tau_i \vdash F_i\sigma$ for $i = 1 \ldots n$ then there exists σ' such that $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$.
2. for all inj_k-event$(ev)^\mu$ occurring in Φ, $\mu(\tau_1, \ldots, \tau_n, \sigma) = \mu(\tau'_1, \ldots, \tau'_n, \sigma')$ implies that $\tau_j = \tau'_j$ for all j such that $F_j = \mathrm{inj}_{k_j}$-event(ev_j) for some k_j, ev_j
3. for all inj_{k_1}-event$(ev_1)^{\mu_1}$ and inj_{k_2}-event$(ev_2)^{\mu_2}$ occurring in Φ, $k_1 = k_2$ implies $\mu_1 = \mu_2$

Remark 14. This notation allows us to talk more precisely of the different elements in Definition 4 that describes when a query is satisfied. Formally, given an initial instrumented configuration $\mathcal{C}_I$ and a correspondence query $\psi = (F_1 \wedge \ldots \wedge F_n \Rightarrow \phi)$, the definition of $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)) \models \psi$ can be rewritten as follows: for all $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$, there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash_i, T, \mathsf{Prem}(\psi, T)) \models (F_1, \ldots, F_n), \Phi$.

Similarly, the definition $(\vdash_{is}, \vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)) \models \psi$ can be rewritten as follows: for all $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$, there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash_{is}, T, \mathsf{Prem}(\psi, T)) \models (F_1, \ldots, F_n), \Phi$. ▶

Remark 15. In the definition of $(\vdash, \mathsf{T}) \models \psi$ for some query ψ, some set of traces T and some satisfaction relation $\vdash$, we can always consider that the annotated formula Φ associated to a trace $T \in \mathsf{T}$ assigns to a fact F occurring in ϕ a partial function μ that is defined only on satisfiable facts. Formally, we can request that if F^{μ} occurs in Φ then for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(\psi, T)$, if $\mu(\tau_1, \ldots, \tau_n, \sigma)$ is defined then there exist σ' such that $T, \mu(\tau_1, \ldots, \tau_n, \sigma) \vdash_i F\sigma'$. Such restriction trivially preserves $(\vdash, \mathsf{T}) \models \psi$. ▶

As previously mentioned, we will prove the soundness of our restrictions in several stages. All these stages will typically consist of mapping a trace from one set of traces to another in a way that preserves the satisfaction of the query. In the following definition, we describe the properties we will ensure every time we create these mappings of traces. In Lemma 17, we show the soundness of our mappings w.r.t. the correspondence query.

Definition 34. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $\vdash_1, \vdash_2$ be two intermediate satisfaction relations on facts. Let two sets of intermediate traces $\mathsf{T}_1, \mathsf{T}_2 \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Let $n_{IO} \in \mathbb{N}$.*

*We say that $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-*mapped by *$(\vdash_2, \mathsf{T}_2)$ when for all $T_1 \in \mathsf{T}_1$, for all finite sets $S \subseteq \{(\tau, F) \in steps(T_1) \times \mathbb{F} \mid T_1, \tau \vdash_1 F\}$, there exist $T_2 \in \mathsf{T}_2$, a mapping γ from $steps(T_2)$ to $steps(T_1)$ and a mapping θ from S to $steps(T_2)$ such that:*

For all $\tau, \tau_1, \tau_2 \in steps(T_2)$

1. *for all $\tau \in steps(T_2)$, for all $F \in \mathbb{F}$, if F is an attacker fact $\mathrm{att}_n(M)$ with $n < n_{IO}$ or a table, message or event fact then $T_2, \tau \vdash_i F$ implies $T_1, \gamma(\tau) \vdash_i F$*
2. *for all $\tau_1, \tau_2 \in steps(T_2)$, $\tau_1 \leq \tau_2$ implies $\gamma(\tau_1) \leq \gamma(\tau_2)$*
3. *for all events $F_1, F_2 \in \mathbb{F}$, if $T_2, \tau_1 \vdash_i F_1$ and $T_2, \tau_2 \vdash_i F_2$ and $\gamma(\tau_1) = \gamma(\tau_2)$ then $\tau_1 = \tau_2$*
4. *for all $(\tau, F) \in S$, $T_2, \theta(\tau, F) \vdash_2 F$*
5. *for all $(\tau_1, F_1), (\tau_2, F_2) \in S$, if F_1, F_2 are both events and $\theta(\tau_1, F_1) = \theta(\tau_2, F_2)$ then $(\tau_1, F_1) = (\tau_2, F_2)$*

We say that $(\vdash_1, \mathsf{T}_1)$ is mapped by $(\vdash_2, \mathsf{T}_2)$ when the condition $n < n_{IO}$ in item 1 is replaced by $n \in \mathbb{N}$ (i.e. the property should hold for all phases which is similar to considering $n_{IO} = \infty$).

Mappings between pairs of satisfaction relations and set of traces is transitive and stable by union.

Lemma 16. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $n_{IO} \in \mathbb{N}$. Let $\vdash_1, \vdash_2, \vdash_3$ be three intermediate satisfaction relations on facts. Let the sets of intermediate traces $\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}'_1, \mathsf{T}'_2, \mathsf{T}_3 \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. We have:*

- *if $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$ and $(\vdash_2, \mathsf{T}_2)$ is n_{IO}-mapped by $(\vdash_3, \mathsf{T}_3)$ then $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_3, \mathsf{T}_3)$*
- *if $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$ and $(\vdash_1, \mathsf{T}'_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}'_2)$ then $(\vdash_1, \mathsf{T}_1 \cup \mathsf{T}'_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2 \cup \mathsf{T}'_2)$.*
- *if $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$ then $(\vdash_1, \mathsf{T}_1)$ is n'_{IO}-mapped by $(\vdash_2, \mathsf{T}'_2)$ for all $\mathsf{T}'_2 \supseteq \mathsf{T}_2$ and all $n'_{IO} \leq n_{IO}$.*

Proof. Proving the stability by union is direct from the definition. The proof of transitivity is slightly more complex.

Consider $T_1 \in \mathsf{T}_1$ and a finite set $S \subseteq \{(\tau, F) \in \mathit{steps}(T_1) \times \mathbb{F} \mid T_1, \tau \vdash_1 F\}$. Since $(\vdash_1, \mathsf{T}_1)$ is mapped by $(\vdash_2, \mathsf{T}_2)$, we know that there exists $T_2 \in \mathsf{T}_2$, a mapping γ_1 from $\mathit{steps}(T_2)$ to $\mathit{steps}(T_1)$ and a mapping θ_1 from S to $\mathit{steps}(T_2)$ that satisfied the properties stated in Definition 34.

Let us define $S' = \{(\theta_1(\tau, F), F) \mid (\tau, F) \in S\}$. By definition of θ, we know that $(\theta_1(\tau, F), F) \in \mathit{steps}(T_2) \times \mathbb{F}$. Moreover, by item 4 of Definition 34, we know that $T_2, \theta(\tau, F) \vdash_2 F$. Hence, we can use the set S' and the trace T_2 to obtain from $(\vdash_2, \mathsf{T}_2)$ being mapped by $(\vdash_3, \mathsf{T}_3)$ that there exist $T_3 \in \mathsf{T}_3$, a mapping γ_2 from $\mathit{steps}(T_3)$ to $\mathit{steps}(T_2)$ and a mapping θ_2 from S' to $\mathit{steps}(T_3)$ that satisfied the properties stated in Definition 34.

Let us define $\gamma = [\tau \mapsto \gamma_1(\gamma_2(\tau)) \mid \tau \in \mathit{steps}(T_3)]$ and $\theta = [(\tau, F) \mapsto \theta_2(\theta_1(\tau, F), F) \mid (\tau, F) \in S]$. We now prove that γ and θ satisfy the desired properties with the trace T_3.

Let $\tau, \tau_1, \tau_2 \in \mathit{steps}(T_3)$.

1. Let $F \in \mathbb{F}$ be an attacker fact $\mathrm{att}_n(M)$ with $n < n_{IO}$ or a table, message, or event fact. If $T_3, \tau \vdash_i F$ then we know from $(\vdash_2, \mathsf{T}_2)$ being mapped by $(\vdash_3, \mathsf{T}_3)$ that $T_2, \gamma_2(\tau) \vdash_i F$. From $(\vdash_1, \mathsf{T}_1)$ being mapped by $(\vdash_2, \mathsf{T}_2)$, we deduce that $T_1, \gamma_1(\gamma_2(\tau)) \vdash_i F$ and so $T_1, \gamma(\tau) \vdash_i F$.

2. Since γ_1 and γ_2 preserve order relations, we deduce that $\tau_1 \leq \tau_2$ implies $\gamma_1(\gamma_2(\tau_1)) \leq \gamma_1(\gamma_2(\tau_1))$ and so $\gamma(\tau_1) \leq \gamma(\tau_2)$

3. Let $F_1, F_2 \in \mathbb{F}$ be two event facts such that $T_3, \tau_1 \vdash_i F_1$ and $T_3, \tau_2 \vdash_i F_2$ and $\gamma(\tau_1) = \gamma(\tau_2)$. Since $(\vdash_2, \mathsf{T}_2)$ being mapped by $(\vdash_3, \mathsf{T}_3)$ (item 1), we deduce that $T_2, \gamma_2(\tau_1) \vdash_i F_1$ and $T_2, \gamma_2(\tau_2) \vdash_i F_2$. Thus with $(\vdash_1, \mathsf{T}_1)$ being mapped by $(\vdash_2, \mathsf{T}_2)$ (item 3), we deduce that $\gamma(\tau_1) = \gamma(\tau_2)$ implies $\gamma_2(\tau_1) = \gamma_2(\tau_2)$. By applying again item 3 with γ_2, we obtain that $\tau_1 = \tau_2$.

4. Let $(\tau, F) \in S$. We already showed that $(\theta_1(\tau, F), F) \in S'$. Hence by item 4 on θ_2, we obtain that $T_3, \theta_2(\theta_1(\tau, F), F) \vdash_3 F$ and so $T_3, \theta(\tau, F) \vdash_3 F$

5. Let $(\tau_1, F_1), (\tau_2, F_2) \in S$ such that F_1, F_2 are both events and $\theta(\tau_1, F_1) = \theta(\tau_2, F_2)$. By definition of θ, we have $\theta_2(\theta_1(\tau_1, F_1), F_1) = \theta_2(\theta_1(\tau_2, F_2), F_2)$. Moreover, since $\theta_1(\tau_1, F_1), \theta_1(\tau_2, F_2) \in S'$, we can apply item 5 on θ_2 to obtain that $\theta_1(\tau_1, F_1) = \theta_1(\tau_2, F_2)$. By applying item 5 on θ_1, we conclude that $\tau_1 = \tau_2$.

This allows us to conclude that $(\vdash_1, \mathsf{T}_1)$ is mapped by $(\vdash_3, \mathsf{T}_3)$. □

We can now show how mappings allow us to prove the soundness of our restrictions.

Lemma 17. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $n_{IO} \in \mathbb{N}$. Let $\psi = (F_1 \wedge \ldots F_n \Rightarrow \phi)$ be a IO-n_{IO}-compliant correspondence query such that $\mathit{names}(\psi) \subseteq \mathit{dom}(\rho_I)$.*

Let $\vdash_1, \vdash_2$ be two intermediate satisfaction relations on facts. Let two sets of intermediate traces $\mathsf{T}_1, \mathsf{T}_2 \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. If $(\vdash_2, \vdash_i, \mathsf{T}_2) \models \psi$ and $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$ then for all $T \in \mathsf{T}_1$, for all finite sets $\mathcal{P} \subseteq \mathsf{Prem}(\psi, T)$, there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash_1, T_1, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$.

Proof. Consider a trace $T_1 \in \mathsf{T}_1$ and a finite set of $\mathcal{P} \subseteq \mathsf{Prem}(\psi, T_1)$. Note that all elements in $\mathcal{P}$ are of the form $(\tau_1, \ldots, \tau_n, \sigma)$ where $\tau_1, \ldots, \tau_n \in \mathit{steps}(T_1)$ and σ is a ground substitution such that $\mathit{dom}(\sigma) = \mathit{vars}(F_1, \ldots, F_n)$. With $\mathcal{P}$ being finite, we can thus build a finite set $S_0 = \{(\tau_1, F_1\sigma); \ldots; (\tau_n, F_n\sigma) \mid (\tau_1, \ldots, \tau_n, \sigma) \in \mathcal{P}\}$. To match the hypotheses of Definition 34, we can remove from S_0 the pairs that do not correspond to the satisfaction of a fact, i.e. we build $S = \{(\tau, F) \in S_0 \mid T_1, \tau \vdash_1 F\}$.

Since $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$, we know by definition that there exist $T_2 \in \mathsf{T}_2$, a mapping γ from $\mathit{steps}(T_2)$ to $\mathit{steps}(T_1)$ and a mapping θ from S to $\mathit{steps}(T_2)$ that satisfy the properties stated in Definition 34.

By hypothesis, we also know that $(\vdash_2, \vdash_i, \mathsf{T}_2) \models \psi$ hence there exists an annotated formula Φ' w.r.t. $(n, \mathsf{max}_{\mathit{step}}(T_2))$ such that $\overline{\Phi'} = \phi$ and $(\vdash_2, T_2, \mathsf{Prem}(\psi, T_2)) \models (F_1, \ldots, F_n), \Phi'$. We build a new annotated formula Φ w.r.t. $(n, \mathsf{max}_{\mathit{step}}(T_1))$ from Φ' by replacing every occurence $F^{\mu'}$ in Φ' by F^{μ} where μ is defined as follows:

- $\mathit{dom}(\mu) \subseteq \mathcal{P}$
- for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathcal{P}$, $\mu(\tau_1, \ldots, \tau_n, \sigma)$ is defined if and only if $\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma)$ is defined
- for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathcal{P}$, if $\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma)$ is defined, then $\mu(\tau_1, \ldots, \tau_n, \sigma) = \gamma(\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma))$.

By construction, $\overline{\Phi} = \overline{\Phi'} = \phi$. Hence, it remains to show that $(\vdash_1, T_1, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$. We prove the three properties of $(\vdash_1, T_1, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$ separately:

1. Let $(\tau_1, \ldots, \tau_n, \sigma) \in \mathcal{P}$. Assume that $T_1, \tau_i \vdash_1 F_i\sigma$ for all $i = 1 \ldots n$. Thus $(\tau_i, F_i\sigma) \in S$ for all $i = 1 \ldots n$. Let us consider $\tau'_1, \ldots, \tau'_n$ such that $\tau'_i = \theta(\tau_i, F_i\sigma)$ for all $i = 1 \ldots n$. By item 4 of Definition 34, we know that $T_2, \theta(\tau_i, F_i\sigma) \vdash_2 F_i\sigma$ for all $i = 1 \ldots n$. Since $(\vdash_2, T_2, \mathsf{Prem}(\psi, T_2)) \models \Phi'$ holds, we deduce that there exists σ' such that $F_i\sigma = F_i\sigma'$ for all $i = 1 \ldots n$ and $(\vdash_i, T_2, (\tau'_1, \ldots, \tau'_n, \sigma)) \models \Phi'\sigma'$.

   We now prove that $(\vdash_i, T_1, (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$. To do so, let us look at any occurrence F^{μ} in Φ. By definition, we know that the corresponding occurrence in Φ' is $F^{\mu'}$ where $\mu(\tau_1, \ldots, \tau_n, \sigma)$ is defined if and only if $\mu'(\tau'_1, \ldots, \tau'_n, \sigma)$ is defined. Moreover, if $(\vdash_i, T_2, (\tau'_1, \ldots, \tau'_n, \sigma)) \models F\sigma'^{\mu'}$ then it implies that $\mu'(\tau'_1, \ldots, \tau'_n, \sigma)$ is defined and $T_2, \mu'(\tau'_1, \ldots, \tau'_n, \sigma) \vdash_i F\sigma'$. Since ψ is IO-n_{IO}-compliant, we know that if F is an attacker fact $\mathrm{att}_i(M)$ then $i < n_{IO}$. Thus, we can apply item 1 of Definition 34 which implies $T_1, \gamma(\mu'(\tau'_1, \ldots, \tau'_n, \sigma)) \vdash_i F\sigma'$. Since $\mu(\tau_1, \ldots, \tau_n, \sigma) = \gamma(\mu'(\tau'_1, \ldots, \tau'_n, \sigma))$, we deduce $(\vdash_i, T_1, (\tau_1, \ldots, \tau_n, \sigma)) \models F\sigma'^{\mu}$.

   Moreover, to cover nested queries, we also need to prove that order between steps are preserved. Thus consider a second occurrence $F''^{\mu''}$ of a fact in Φ' such that $\mu''(\tau'_1, \ldots, \tau'_n, \sigma)$ is defined and $\mu'(\tau'_1, \ldots, \tau'_n, \sigma) \leq \mu''(\tau'_1, \ldots, \tau'_n, \sigma)$. Once again by construction of Φ, we know that the corresponding occurrence of $F''^{\mu''}$ in Φ is the annotated fact $F''^{\mu'''}$ where $\mu'''(\tau_1, \ldots, \tau_n, \sigma) = \gamma(\mu''(\tau'_1, \ldots, \tau'_n, \sigma))$. By item 2 of Definition 34, $\mu'(\tau'_1, \ldots, \tau'_n, \sigma) \leq \mu''(\tau'_1, \ldots, \tau'_n, \sigma)$ implies $\gamma(\mu'(\tau'_1, \ldots, \tau'_n, \sigma)) \leq \gamma(\mu''(\tau'_1, \ldots, \tau'_n, \sigma))$ and so $\mu(\tau_1, \ldots, \tau_n, \sigma) \leq \mu'''(\tau_1, \ldots, \tau_n, \sigma)$.

   This conclude the proof of $(\vdash_i, T_1, (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$.

2. Consider an injective event $\text{inj}_k\text{-event}(ev)^\mu$ occurring in Φ such that $\mu(\tau_1, \ldots, \tau_n, \sigma) = \mu(\tau'_1, \ldots, \tau'_n, \sigma')$. By definition of Φ, we know that the same occurrence of the injective event in Φ' is $\text{inj}_k\text{-event}(ev)^{\mu'}$ where $\mu(\tau_1, \ldots, \tau_n, \sigma) = \gamma(\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma))$ and $\mu(\tau'_1, \ldots, \tau'_n, \sigma') = \gamma(\mu'(\theta(\tau'_1, F_1\sigma'), \ldots, \theta(\tau'_n, F_n\sigma), \sigma'))$. Since $\mu(\tau_1, \ldots, \tau_n, \sigma)$ is defined then both $\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma)$ and $\mu'(\theta(\tau'_1, F_1\sigma'), \ldots, \theta(\tau'_n, F_n\sigma), \sigma')$ are defined meaning that there exist σ'', σ''' such that $T_2, \mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma) \vdash_i \text{inj}_k\text{-event}(ev)\sigma''$ and $T_2, \mu'(\theta(\tau'_1, F_1\sigma'), \ldots, \theta(\tau'_n, F_n\sigma), \sigma') \vdash_i \text{inj}_k\text{-event}(ev)\sigma'''$. Thus by item 3 of Definition 34, we deduce that $\mu'(\theta(\tau_1, F_1\sigma), \ldots, \theta(\tau_n, F_n\sigma), \sigma) = \mu'(\theta(\tau'_1, F_1\sigma'), \ldots, \theta(\tau'_n, F_n\sigma), \sigma')$. But since $(\vdash_2, \mathsf{T}_2) \models \psi$, we deduce that $\theta(\tau_j, F_j\sigma) = \theta(\tau'_j, F_j\sigma')$ for all j such that F_j is an injective event. Thus, by item 5 of Definition 34, we conclude that $\tau_j = \tau'_j$ and $F_j\sigma = F_j\sigma'$, which allows us to conclude.

3. The third property is direct by construction of Φ.

This concludes the proof of $(\vdash_1, T_1, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$. □

Lemma 17 does not allow us to directly prove that a correspondence query is satisfied by the set of traces T_1. Indeed, the lemma tells us that for all traces T of T_1, for all *finite* subsets of $\mathcal{P} \subseteq \mathsf{Prem}(\psi, T)$, there exists an annotated formula Φ that verifies the correspondence query, i.e. $(\vdash_1, T_1, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$. However, to show that T satisfies the correspondence query ψ, we need to show that there exists an annotated formula Φ such that $(\vdash_1, T_1, \mathsf{Prem}(\psi, T)) \models (F_1, \ldots, F_n), \Phi$. Ideally, we would like to take $\mathcal{P} = \mathsf{Prem}(\psi, T)$ and apply Lemma 17 to conclude. But this is impossible since $\mathsf{Prem}(\psi, T)$ is infinite and $\mathcal{P}$ must be finite in Lemma 17. The next lemma solves this issue.

Lemma 18. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $\psi = (F_1 \wedge \ldots \wedge F_n \Rightarrow \phi)$ be a correspondence query such that $names(\psi) \subseteq dom(\rho_I)$. Let $\mathsf{T} \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Let $\vdash$ be an intermediate satisfaction relation on facts.*

If for all $T \in \mathsf{T}$, for all finite sets $\mathcal{P} \subseteq \mathsf{Prem}(\psi, T)$, there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash, T, \mathcal{P}) \models (F_1, \ldots, F_n), \Phi$ then $(\vdash, \vdash_i, \mathsf{T}) \models \psi$.

Proof. Consider a trace $T \in \mathsf{T}$. In this proof, we annotate all injective events of ϕ from 1 to n_{iev} where n_{iev} is the number of injective events in ϕ. Thus, an injective event occurring in ϕ will be denoted $\text{inj}_k\text{-event}(ev)^i$. Similarly, we also annotate all facts in ϕ different from an injective event from 1 to n_{fact} where n_{fact} is the number of facts in ϕ different from an injective event. For an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$, we also annotate the injective events and facts different from injective events in a similar fashion, meaning that an injective event occurring in Φ will be denoted $\text{inj}_k\text{-event}(ev)^{i,\mu}$ where $i \in \{1, \ldots, n_{iev}\}$ and μ is the partial function from $\mathsf{Prem}(\psi, T)$ to $steps(T)$ as defined in Section 2.3.1. These annotations allow us to more easily talk about a specific occurrence of an injective event in ϕ or an annotated formula Φ. Finally, given an annotated formula Φ, we define the functions α_Φ and β_Φ such that for all $i \in \{1, \ldots, n_{iev}\}$, $\alpha_\Phi(i) = \mu$ where $\text{inj}_k\text{-event}(ev)^\mu_i$ occurs in Φ for some ev, k; and for all $i \in \{1, \ldots, n_{fact}\}$, $\beta_\Phi(i) = \mu$ where $F^{i,\mu}$ occurs in Φ for some fact different from an injective event.

Finally, given $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$, we denote by $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ the largest set of tuples $(i_1, \ldots, i_{n_{iev}})$ such that $(i_1, \ldots, i_{n_{iev}}) \in \mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ when there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$, $(\vdash, T, \{(\tau_1, \ldots, \tau_n, \sigma)\}) \models (F_1, \ldots, F_n), \Phi$ and for all $k \in \{1, \ldots, n_{iev}\}$,

- if $\alpha_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma)$ is defined then $\alpha_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma) = i_k$; and
- if $\alpha_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma)$ is not defined then $i_k = 0$.

Intuitively, $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ represents all the possible ways to match the injective events from the query to the trace in order to satisfy the query. Note that by definition of annotated formulae, $(i_1, \ldots, i_{iev}) \in \mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ implies $0 \leq i_k \leq \mathsf{max}_{step}(T)$ for all k. Thus, for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$, $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ is finite. Moreover, by our hypothesis, we know that there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash, T, \{(\tau_1, \ldots, \tau_n, \sigma)\}) \models (F_1, \ldots, F_n), \Phi$. Therefore, for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$, $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma)$ is finite and non empty.

We define the binary relation $\sim$ on $\mathsf{Prem}(T, \psi)$ such that $(\tau_1, \ldots, \tau_n, \sigma) \sim (\tau'_1, \ldots, \tau'_n, \sigma')$ if and only if $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma) = \mathsf{Sol}(\tau'_1, \ldots, \tau'_n, \sigma')$ and $\tau_j = \tau'_j$ for all $j \in \{1, \ldots, n\}$. Notice that the quotient set $\mathsf{Prem}(T, \psi)/\sim$ is finite (to be precise it contains at most $2^{(\mathsf{max}_{step}(T)+1)^{n_{iev}}} \times (\mathsf{max}_{step}(T)+1)^n$ elements). We consider the set Q defined by taking a representative of each equivalence class in $\mathsf{Prem}(T, \psi)/\sim$.

Since $\mathsf{Prem}(T, \psi)/\sim$ is finite then Q is also finite, meaning that we can applying our hypothesis on Q. This allows us to deduce that there exists an annotated formula Φ_0 w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi_0} = \phi$ and $(\vdash, T, Q) \models (F_1, \ldots, F_n), \Phi_0$.

For all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$, if $(\tau_1, \ldots, \tau_n, \sigma') \in Q$ such that $(\tau_1, \ldots, \tau_n, \sigma) \sim (\tau_1, \ldots, \tau_n, \sigma')$ (the representative exists and is unique) then we know from $(\vdash, T, Q) \models (F_1, \ldots, F_n), \Phi_0$ that there exists $(i_1, \ldots, i_{iev}) \in \mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma')$ such that $\alpha_{\Phi_0}(k)(\tau_1, \ldots, \tau_n, \sigma') = i_k$ for all $k \in \{1, \ldots, n_{iev}\}$. Moreover, since $\mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma) = \mathsf{Sol}(\tau_1, \ldots, \tau_n, \sigma')$, we also know that there exists an annotated formula Φ_1 such that $(\vdash, T, \{(\tau_1, \ldots, \tau_n, \sigma)\}) \models (F_1, \ldots, F_n), \Phi_1$ and $\alpha_{\Phi_1}(k)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_0}(k)(\tau_1, \ldots, \tau_n, \sigma')$ for all $k \in \{1, \ldots, n_{iev}\}$. Let us denote the annotated formula Φ_1 by $AF(\tau_1, \ldots, \tau_n, \sigma)$.

We can now build that annotated formulae Φ that will cover completely $\mathsf{Prem}(T, \psi)$ as the annotated formula that satisfies the following property: For all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$, if $\Phi' = AF(\tau_1, \ldots, \tau_n, \sigma)$ then

- for all $k \in \{1, \ldots, n_{iev}\}$, $\alpha_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi'}(k)(\tau_1, \ldots, \tau_n, \sigma)$
- for all $k \in \{1, \ldots, n_{fact}\}$, $\beta_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma) = \beta_{\Phi'}(k)(\tau_1, \ldots, \tau_n, \sigma)$

Intuitively, the value of the partial function on $(\tau_1, \ldots, \tau_n, \sigma)$ for an occurrence of a fact (injective or not) in Φ is the same as the value of the partial function on $(\tau_1, \ldots, \tau_n, \sigma)$ for the *same* occurrence in $AF(\tau_1, \ldots, \tau_n, \sigma)$.

It remains to prove that $(\vdash, T, \mathsf{Prem}(\psi, T)) \models (F_1, \ldots, F_n), \Phi$. We consider the three items of the definition separately:

1. Let $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$. Assume that $T, \tau_i \vdash_i F_i\sigma$ for all $i = 1 \ldots n$. By definition, we know that $(\vdash, T, \{(\tau_1, \ldots, \tau_n, \sigma)\}) \models (F_1, \ldots, F_n), \Phi_1$ where $\Phi_1 = AF(\tau_1, \ldots, \tau_n, \sigma)$. Therefore, there exists σ' such that $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi_1\sigma'$. Since for all $k \in \{1, \ldots, n_{iev}\}$, $\alpha_\Phi(k)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_1}(k)(\tau_1, \ldots, \tau_n, \sigma)$ and for all $k' \in \{1, \ldots, n_{fact}\}$, $\beta_\Phi(k')(\tau_1, \ldots, \tau_n, \sigma) = \beta_{\Phi_1}(k')(\tau_1, \ldots, \tau_n, \sigma)$, we deduce that $(\vdash_i, T, (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$.

2. Let inj_k-$\mathrm{event}(ev)^{i,\mu}$ occurring in Φ, i.e. $i \in \{1, \ldots, n_{iev}\}$. Assume that $\mu(\tau_1, \ldots, \tau_n, \sigma) = \mu(\tau'_1, \ldots, \tau'_n, \sigma')$. By definition of α_Φ, $\mu = \alpha_\Phi(i)$. Hence, $\alpha_\Phi(i)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_\Phi(i)(\tau'_1,$

$\ldots, \tau'_n, \sigma')$. Moreover, by definition of Φ, we know that $\alpha_\Phi(i)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_1}(i)(\tau_1, \ldots, \tau_n, \sigma)$ with $\Phi_1 = AF(\tau_1, \ldots, \tau_n, \sigma)$. By definition of $AF(\tau_1, \ldots, \tau_n, \sigma)$, we deduce that there exists σ'' such that $(\tau_1, \ldots, \tau_n, \sigma'') \in Q$ and $\alpha_{\Phi_1}(i)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_0}(i)(\tau_1, \ldots, \tau_n, \sigma'')$ and so $\alpha_\Phi(i)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_0}(i)(\tau_1, \ldots, \tau_n, \sigma'')$. Similarly, there exists σ''' such that $(\tau'_1, \ldots, \tau'_n, \sigma''') \in Q$ such that $\alpha_\Phi(i)(\tau'_1, \ldots, \tau'_n, \sigma') = \alpha_{\Phi_0}(i)(\tau'_1, \ldots, \tau'_n, \sigma''')$.

Therefore, we obtain that $\alpha_{\Phi_0}(i)(\tau_1, \ldots, \tau_n, \sigma'') = \alpha_{\Phi_0}(i)(\tau'_1, \ldots, \tau'_n, \sigma''')$. Since $(\vdash, T, Q) \models (F_1, \ldots, F_n), \Phi_0$ and $(\tau_1, \ldots, \tau_n, \sigma'') \in Q$ and $(\tau'_1, \ldots, \tau'_n, \sigma''') \in Q$, we can conclude that $\tau_j = \tau'_j$ for all $j \in \{1, \ldots, n\}$ such that F_j is an injective event.

3. Let two injective events $\text{inj}_{k_1}\text{-event}(ev_1)^{i_1,\mu_1}$ and $\text{inj}_{k_2}\text{-event}(ev_2)^{i_2,\mu_2}$ occurring in Φ with $k_1 = k_2$. We need to show that $\mu_1 = \mu_2$, i.e. $\alpha_\Phi(i_1) = \alpha_\Phi(i_2)$. Since $(\vdash, T, Q) \models (F_1, \ldots, F_n), \Phi_0$, we know that $\alpha_{\Phi_0}(i_1) = \alpha_{\Phi_0}(i_2)$. Let $(\tau_1, \ldots, \tau_n, \sigma) \in \mathsf{Prem}(T, \psi)$. By definition of Φ, we know $\alpha_\Phi(i_1)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_1}(i_1)(\tau_1, \ldots, \tau_n, \sigma)$ and $\alpha_\Phi(i_2)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_1}(i_2)(\tau_1, \ldots, \tau_n, \sigma)$ where $\Phi_1 = AF(\tau_1, \ldots, \tau_n, \sigma)$. Moreover, by definition of $AF(\tau_1, \ldots, \tau_n, \sigma)$, we know there exists σ' such that $(\tau_1, \ldots, \tau_n, \sigma') \in Q$, $\alpha_{\Phi_1}(i_1)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_0}(i_1)(\tau_1, \ldots, \tau_n, \sigma')$ and $\alpha_{\Phi_1}(i_2)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_{\Phi_0}(i_2)(\tau_1, \ldots, \tau_n, \sigma')$. Since $\alpha_{\Phi_0}(i_1) = \alpha_{\Phi_0}(i_2)$, we deduce that $\alpha_\Phi(i_1)(\tau_1, \ldots, \tau_n, \sigma) = \alpha_\Phi(i_2)(\tau_1, \ldots, \tau_n, \sigma)$. Hence we conclude that $\alpha_\Phi(i_1) = \alpha_\Phi(i_2)$.

This concludes the proof that for all traces $T \in \mathsf{T}$, there exists an annotated formula Φ w.r.t. $(n, \mathsf{max}_{step}(T))$ such that $\overline{\Phi} = \phi$ and $(\vdash, T, \mathsf{Prem}(\psi, T)) \models (F_1, \ldots, F_n), \Phi$. Therefore, we conclude that $(\vdash, \vdash_i, \mathsf{T}) \models \psi$. □

By direct application of Lemmas 17 and 18, we obtain the following corollary.

Corollary 1. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $n_{IO} \in \mathbb{N}$. Let $\psi = (F_1 \wedge \ldots F_n \Rightarrow \phi)$ be an IO-n_{IO}-compliant correspondence query such that $names(\psi) \subseteq dom(\rho_I)$.*

Let $\vdash_1, \vdash_2$ be two intermediate satisfaction relations on facts. Let two sets of intermediate traces $\mathsf{T}_1, \mathsf{T}_2 \subseteq \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. If $(\vdash_2, \vdash_i, \mathsf{T}_2) \models \psi$ and $(\vdash_1, \mathsf{T}_1)$ is n_{IO}-mapped by $(\vdash_2, \mathsf{T}_2)$ then $(\vdash_1, \vdash_i, \mathsf{T}_1) \models \psi$.

A.1 $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is mapped by $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$

Lemma 19. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is mapped by $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$.*

Proof. Let $T_1 \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. We need to prove that for all finite sets $S \subseteq \{(\tau, F) \in steps(T_1) \times \mathbb{F} \mid T_1, \tau \vdash_i F\}$, there exist $T_2 \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$, a mapping γ from $steps(T_2)$ to $steps(T_1)$ and a mapping θ from S to $steps(T_2)$ that satisfy the properties stated in Definition 34.

We prove this result by induction on $|S|$. However, for our inductive hypothesis, we need to reenforce the properties required on θ. Let us denote $\mathsf{S}_{all} = \{(\tau, F) \in steps(T_1) \times \mathbb{F} \mid T_1, \tau \vdash_i F\}$. First, instead of θ being a mapping from S to $steps(T_2)$, we ask θ to be a mapping from S_{all} to $steps(T_2)$. Second, item 4 remains unchanged on S (i.e. for all $(\tau, F) \in S$, $T_2, \theta(\tau, F) \vdash_{is} F$) but is extended to the whole domain of θ as follows: for all $(\tau, F) \in \mathsf{S}_{all}$, $T_2, \theta(\tau, F) \vdash_i F$. Note this additional condition is not in conflict when restricted to S since

$T_2, \theta(\tau, F) \vdash_{is} F$ implies $T_2, \theta(\tau, F) \vdash_i F$. Third, item 5 applies not only to S but to the whole set S_{all}, i.e. for all $(\tau_1, F_1), (\tau_2, F_2) \in \mathsf{S}_{all}$, if F_1, F_2 are both events and $\theta(\tau_1, F_1) = \theta(\tau_2, F_2)$ then $(\tau_1, F_1) = (\tau_2, F_2)$.

Moreover, in the rest of the proof, we will say items 4 and 5 when talking about the corresponding reenforced properties stated above.

Base case $|S| = 0$*:* We directly conclude by taking $T_2 = T_1$, the identity mapping for γ and the mapping $[(\tau, F) \mapsto \tau \mid (\tau, F) \in \mathsf{S}_{all}]$ for θ.

Inductive step $|S| > 0$*:* In such a case, $S = \{(\tau_0, F_0)\} \cup S'$ for some τ_0, F_0, S'. By applying our inductive hypothesis on S', we obtain a trace $T_2' \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$, a mapping γ' from $steps(T_2')$ to $steps(T_1)$ and a mapping θ' from S_{all} to $steps(T_2')$ that satisfy the inductive properties.

Let us do a case analysis on F_0:

- Case $F_0 = \text{event}(o, ev)$ or $F_0 = \text{inj}_k\text{-event}(ev)$ or $F_0 = \text{msg}_i(M, N)$: by definition of $\vdash_{is}$, $T_2', \tau_0 \vdash_i F_0$ if and only if $T_2', \tau_0 \vdash_{is} F_0$.

  Let us consider $T_2 = T_2'$ and $\gamma = \gamma'$. In such a case, items 1, 2 and 3 directly hold since γ' satisfies the inductive properties. Moreover, by taking $\theta = \theta'$, we also deduce that item 4 and 5 hold for θ. Indeed, by definition of S, we know that $T_1, \tau_0 \vdash_i F_0$. Hence since θ' satisfies item 4 on S' and T_2', we have $T_2', \theta(\tau_0, F_0) \vdash_i F_0$. Finally, since $T_2 = T_2'$ and $T_2', \tau_0 \vdash_i F_0$ if and only if $T_2', \tau \vdash_{is} F_0$, we conclude.

- Case $F_0 = \text{table}_i(tbl(M_1', \ldots, M_m'))$: Let us look at the configuration $T_2'[\mathsf{max}_{step}(T_2')] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$. If $n < i$ then we consider the trace on which we apply $i - n$ times the phase rule I-Phase at the end of T_2', i.e. $T_2 = T_2'[0] \rightarrow_i^* T_2'[\mathsf{max}_{step}(T_2')] \rightarrow_i^* i, \rho, \mathcal{P}', \mathcal{T}, \mathcal{A}, \Lambda$ for some $\mathcal{P}'$. If $n \geq i$, then we consider $T_2 = T_2'$.

  In both cases, we define γ as the mapping $\gamma'[\mathsf{max}_{step}(T_2') + j \mapsto \mathsf{max}_{step}(T_2') \mid j \in \{1, \ldots, i - n\}]$. (When $n \geq i$, this definition gives us $\gamma = \gamma'$.) Intuitively, all the steps that we added at the end of T_2' in T_2 are mapped to $\mathsf{max}_{step}(T_2')$. First, by construction, we trivially have that γ satisfies item 2 since it holds for γ'. Second, since we only added possibly some instances of the phase I-Phase, we deduce that that for all facts F, for all steps τ, if $T_2, \tau, \vdash_i F$ and F is an event, injective event or message predicates then $\tau \leq \mathsf{max}_{step}(T_2')$. Thus, item 3 holds for γ since it holds for γ'. Third, if $T_2, \tau \vdash_i F$ then either $\tau \leq \mathsf{max}_{step}(T_2')$ and in such a case we have $\gamma(\tau) = \gamma'(\tau)$ and $T_2, \gamma'(\tau, F) \vdash_i F$, or $\tau > \mathsf{max}_{step}(T_2')$ and F is necessarily an attacker and table lookup predicates since we only added some instances of the phase rule I-Phase. In such a case, by definition of $\vdash_i$, we deduce that $T_2, \mathsf{max}_{step}(T_2') \vdash_i F$. This allows us to conclude that item 1 holds for γ.

  We now define θ. First, for all $(\tau, F) \in \mathsf{S}_{all} \setminus \{(\tau_0, F_0)\}$, we define $\theta(\tau, F) = \theta'(\tau, F)$. For (τ_0, F_0), we need to find a step τ_0' such that $T_2, \tau_0' \vdash_{is} F_0$. By definition of S, $T_1, \tau_0 \vdash_i F_0$. Since item 4 holds for θ', we deduce that $T_2', \theta'(\tau_0, F_0) \vdash_i F_0$. Thus there exists $tbl(M_1, \ldots, M_n) \in \mathcal{T}(T_2'[\theta'(\tau_0, F_0)])$ such that $tbl(M_1, \ldots, M_n)\rho = tbl(M_1', \ldots, M_n')$. Since the semantics rules ensure that the set of elements inserted in tables can only grow and by construction of T_2 (i.e. the phase at step $\mathsf{max}_{step}(T_2)$ in T_2 is greater or equal to i), we deduce that there exists $\tau_0' \geq \theta'(\tau_0, F_0)$ such that $T_2, \tau_0' \vdash_{is} F_0$. Thus, we define $\theta(\tau_0, F_0) = \tau_0'$. By construction, we obtain that θ satisfies item 4. Moreover, since F_0 is not an event fact, we directly deduce that item 5 holds for θ since it holds for θ'.

- Case $F_0 = \mathrm{att}_i(M')$: Since θ' satisfies item 4, we deduce that $T_2', \theta'(\tau_0, F_0) \vdash_i F_0$. Let us denote $\tau_0' = \theta'(\tau_0, F_0)$. By definition of $\vdash_i$, we know that there exist M and $T_2'[\tau_0'] \to_i^* \mathcal{C} = i, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ only by the rules I-App, I-New and I-Phase such that $M' = M\rho$ and $M \in \mathcal{A}$. W.l.o.g., we can assume that $T_2'[\tau_0'] \to_i^* \mathcal{C}' \to_i^* \mathcal{C}$ where $T_2'[\tau_0'] \to_i^* \mathcal{C}'$ consists of k applications of only the rules I-App and I-New (k possibly being 0), and $\mathcal{C}' \to_i^* \mathcal{C}$ consists only of applications of the rule I-Phase. Note that since the rules I-App and I-New only increase the attacker knowledge, we have that $\mathcal{A} = T_2'[\tau_0'] \cup \mathcal{A}_M$ for some $\mathcal{A}_M$.

  We build T_2 by *inserting* the rules applied in $T_2'[\tau_0'] \to_i^* \mathcal{C}'$ in the trace T_2' between the transitions $T_2'[\tau_0']$ and $T_2'[\tau_0' + 1]$, and possibly adding some rule Phase at the end of the trace if the phase of $T_2'[\mathsf{max}_{step}(T')]$ is strictly smaller then i (similarly to the case where F_0 was a table lookup predicate). Formally, we build the following trace T_2:

$$T_2'[0] \xrightarrow{\ell_1}_i T_2'[1] \xrightarrow{\ell_2}_i \dots \xrightarrow{\ell_{\tau_0'}}_i T_2'[\tau_0'] \to_i^* \mathcal{C}' \xrightarrow{\ell_{\tau_0'+1}}_i \mathcal{C}_1 \xrightarrow{\ell_{\tau_0'+2}}_i \dots \xrightarrow{\ell_{\tau_0'+N}}_i \mathcal{C}_N \to_i^* \mathcal{C}''$$

  where $N = \mathsf{max}_{step}(T_2') - \tau_0'$, only the rule I-Phase is applied in $\mathcal{C}_N \to_i^* \mathcal{C}''$, the phase of $\mathcal{C}''$ is greater than i and for all $j \in \{1, \dots, N\}$, if $T_2'[\tau_0' + j] = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}', \Lambda'$ then $\mathcal{C}_j = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}' \cup \mathcal{A}_M, \Lambda'$.

  By construction and more specifically since we only added instances of I-App, I-New and I-Phase rules, we obtain that for all $F \in \mathbb{F}$,

  - for all $\tau \leq \tau_0'$, $T_2', \tau \vdash_i F$ if and only if $T_2, \tau \vdash_i F$; and $T_2', \tau \vdash_{is} F$ implies $T_2, \tau \vdash_{is} F$
  - for all $\mathsf{max}_{step}(T_2') \geq \tau > \tau_0'$, $T_2', \tau \vdash_i F$ if and only if $T_2, \tau + k \vdash_i F$; and $T_2', \tau \vdash_{is} F$ implies $T_2, \tau + k \vdash_{is} F$.
  - for all $\mathsf{max}_{step}(T_2) \geq \tau > \mathsf{max}_{step}(T_2') + k$, $T_2, \tau \vdash_i F$ implies $T_2', \mathsf{max}_{step}(T_2') \vdash_i F$
  - if F is an event fact and $T_2, \tau \vdash_i F$ then $\tau \leq \tau_0'$ or $\mathsf{max}_{step}(T_2') + k \geq \tau > \tau_0' + k$.
  - since $M \in \mathcal{A}$ (and so $M \in \mathcal{A}(T[\tau_0' + k + j])$ for all $j > 0$) and since the phase of $\mathcal{C}''$ is greater than i, we know that there exists a step $\tau_1 \geq \tau_0' + k$ such that $T_2, \tau_1 \vdash_{is} F_0$.

  We conclude by building γ and θ as follows:

$$\gamma(\tau) = \begin{cases} \gamma'(\tau) & \text{if } \tau \leq \tau_0' \\ \gamma'(\tau_0') & \text{if } \tau_0' < \tau \leq \tau_0' + k \\ \gamma'(\tau - k) & \text{if } \tau_0' + k < \tau \leq \mathsf{max}_{step}(T_2') + k \\ \gamma'(\mathsf{max}_{step}(T_2')) & \text{if } \mathsf{max}_{step}(T_2') + k < \tau \end{cases}$$

$$\theta(\tau, F) = \begin{cases} \tau_1 & \text{if } (\tau, F) = (\tau_0, F_0) \\ \theta'(\tau, F) & \text{if } (\tau, F) \in \mathsf{S}_{all} \setminus \{(\tau_0, F_0)\} \text{ and } \theta'(\tau, F) \leq \tau_0' \\ \theta'(\tau, F) + k & \text{if } (\tau, F) \in \mathsf{S}_{all} \setminus \{(\tau_0, F_0)\} \text{ and } \theta'(\tau, F) > \tau_0' \end{cases}$$

□

A.2 $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \to_i))$ is n_{IO}-mapped by $(\vdash_{is}, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i))$

In the traces of $\mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i)$, we consider our two main restrictions, that are data compliance and IO-n_{IO}-compliance. To show the mapping of $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \to_i))$ by $(\vdash_{is}, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i))$, we will need to transform the traces by applying several instances of the I-App, I-Msg,

I-Out and I-In rules. Thanks to Lemma 16, we know that our mappings are transitive and stable by union, meaning that we can show first show mapping for our very small transformations and then combine them by multiple instance of Lemma 16.

Lemma 20. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Let $\tau_0 \in steps(T)$. Consider the trace T' build by inserting an application of the* I-App *rule at the step τ_0. Formally, T' is defined as follows:*

$$T[0] \xrightarrow{\ell_1}_i T[1] \xrightarrow{\ell_2}_i \ldots \xrightarrow{\ell_{\tau_0}}_i T[\tau_0] \xrightarrow{\text{I-App}(f,M_1,\ldots,M_m)}_i \mathcal{C}_0 \xrightarrow{\ell_{\tau_0+1}}_i \mathcal{C}_1 \xrightarrow{\ell_{\tau_0+2}}_i \ldots \xrightarrow{\ell_{\tau_0+N}}_i \mathcal{C}_N$$

where $N = \mathsf{max}_{step}(T) - \tau_0$, the application of $T[\tau_0] \xrightarrow{\text{I-App}(f,M_1,\ldots,M_m)}_i \mathcal{C}$ yields the term M (i.e. $f(M_1,\ldots,M_m) \Downarrow M$) and for all $j \in \{0,\ldots,N\}$, if $T[\tau_0+j] = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}', \Lambda'$ then $\mathcal{C}_j = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}' \cup \{M\}, \Lambda'$.

We have that $(\vdash_{is}, \{T\})$ is mapped by $(\vdash_{is}, \{T'\})$.

Proof. Let S be a finite set such that $S \subseteq \{(\tau, F) \in steps(T) \times \mathbb{F} \mid T, \tau \vdash_{is} F\}$. Let us define the following mapping γ and θ:

For all $\tau \in steps(T')$,

$$\gamma(\tau) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 \\ \tau - 1 & \text{if } \tau > \tau_0 \end{cases}$$

For all $(\tau, F) \in S$,

$$\theta(\tau, F) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 \\ \tau + 1 & \text{if } \tau > \tau_0 \end{cases}$$

Let us now prove that γ and θ both satisfies the properties stated in Definition 34. First notice that item 2 trivially holds. Moreover, for all $\tau \in steps(T')$, for all $F \in \mathsf{F}$, if $T', \tau \vdash_i F$ and $\tau \leq \tau_0$ then we trivially have that $T, \gamma(\tau) \vdash_i F$. Otherwise, if $T', \tau \vdash_i F$ and $\tau > \tau_0$ then we know that $T'[\tau]$ is the same configuration as $T[\tau-1]$ except that $\mathcal{A}(T'[\tau]) = \mathcal{A}(T[\tau-1]) \cup \{M\}$. Hence, if F is an attacker fact $\mathrm{att}_n(N')$ then $T', \tau \vdash_i F$ implies $T'[\tau] \rightarrow_i^* \mathcal{C} = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}', \Lambda'$ with $N' = N\rho'$ and $N \in \mathcal{A}'$. Since $M_1, \ldots, M_m \in \mathcal{A}(T[\tau_0])$ and $\tau > \tau_0$, we have $T[\tau - 1] \xrightarrow{\text{I-App}(f,M_1,\ldots,M_m)}_i T[\tau] \rightarrow_i^* \mathcal{C}$ which allow us to conclude that $T, \tau - 1 \vdash_i F$. When F is not an attacker fact, the result directly holds since for all $\tau > \tau_0$, $\mathcal{T}(T'[\tau]) = \mathcal{T}(T[\tau - 1])$ and if $T'[\tau - 1] \xrightarrow{\ell}_i T'[\tau]$ with ℓ an event or message label then $T[\tau - 2] \xrightarrow{\ell}_i T[\tau - 1]$. This allow us to conclude that item 1 holds. For item 3, it suffices to notice that the only case where $\gamma(\tau_1) = \gamma(\tau_2)$ and $\tau_1 \neq \tau_2$ is when $\tau_1 = \tau_0$ and $\tau_2 = \tau_0 + 1$. But no event fact F satisfies $T, \tau_0 + 1 \vdash_i F$ hence the result holds.

The properties on θ are proved in a similar way: For all $(\tau, F) \in S$, if $\tau \leq \tau_0$ then we directly have that $T', \tau \vdash_{is} F$. Moreover, if $\tau > \tau_0$ then we know that $T[\tau - 1] \xrightarrow{\ell}_i T[\tau]$ implies $T'[\tau] \xrightarrow{\ell}_i T'[\tau + 1]$ with $T'[\tau + 1]$ being the same configuration as $T[\tau]$ except that $\mathcal{A}(T'[\tau + 1]) = \mathcal{A}(T[\tau]) \cup \{M\}$. Thus, $T, \tau \vdash_{is} F$ implies $T', \tau + 1 \vdash_{is} F$ and so item 4 holds. Finally, for item 5, it suffices to notice that if $T, \tau \vdash_{is} F_1$ and $T, \tau \vdash_{is} F_2$ with F_1, F_2 two events facts then $F_1 = F_2$. Hence if $\theta(\tau_1, F_1) = \theta(\tau_2, F_2)$ then $\tau_1 = \tau_2$ by construction of θ and so $F_1 = F_2$. □

Lemma 21. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $n_{IO} \in \mathbb{N}$. Let $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Let $\tau_0 \in steps(T)$ such that $T[\tau_0] \xrightarrow{\mathrm{msg}(N[],M)}_i T[\tau_0 + 1]$ by application of the rule* I-I/O *such that $N \in \mathcal{A}_I$ and the phase of $T[\tau_0]$ is $n \geq n_{IO}$. Consider the trace*

T' build by replacing the application of the rule I-I/O *by an application of the rule* I-Out *followed by an application of the rule* I-In *(this is allowed since $N \in \mathcal{A}_I$). Formally, T' is defined as follows:*

$$T[0] \xrightarrow{\ell_1}_i T[1] \xrightarrow{\ell_2}_i \ldots \xrightarrow{\ell_{\tau_0}}_i T[\tau_0] \xrightarrow{\mathrm{msg}(N[],M)}_i \mathcal{C} \xrightarrow{\mathrm{msg}(N[],M)}_i \mathcal{C}_1 \xrightarrow{\ell_{\tau_0+2}}_i \ldots \xrightarrow{\ell_{\tau_0+k}}_i \mathcal{C}_k$$

where $k = \mathsf{max}_{step}(T) - \tau_0$, the transition $T[\tau_0] \xrightarrow{\mathrm{msg}(N[],M)}_i \mathcal{C}$ is an application of the I-Out *rule, the transition $\mathcal{C} \xrightarrow{\mathrm{msg}(N[],M)}_i \mathcal{C}_1$ is an application of the* I-In *rule and for all $j \in \{1, \ldots, k\}$, if $T[\tau_0 + j] = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}', \Lambda'$ then $\mathcal{C}_j = i', \rho', \mathcal{P}', \mathcal{T}', \mathcal{A}' \cup \{M'\}, \Lambda'$ with $M'\rho' = M$.*

We have that $(\vdash_{is}, \{T\})$ is n_{IO}-mapped by $(\vdash_{is}, \{T'\})$.

Proof. Let S be a finite set such that $S \subseteq \{(\tau, F) \in steps(T) \times \mathbb{F} \mid T, \tau \vdash_{is} F\}$. Let us define the following mapping γ and θ:

For all $\tau \in steps(T')$,

$$\gamma(\tau) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 \\ \tau_0 + 1 & \text{if } \tau \in \{\tau_0 + 1; \tau_0 + 2\} \\ \tau - 1 & \text{if } \tau > \tau_0 + 2 \end{cases}$$

For all $(\tau, F) \in S$,

$$\theta(\tau, F) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 \\ \tau + 1 & \text{if } \tau > \tau_0 \end{cases}$$

Let us now prove that γ and θ both satisfies the properties stated in Definition 34. First notice that item 2 trivially holds. Moreover, for all $\tau \in steps(T')$, for all $F \in \mathsf{F}$ such that F is not an attacker fact $\mathrm{att}_i(L)$ with $i \geq n$, if $T', \tau \vdash_i F$ and $\tau \leq \tau_0$ then we trivially have that $T, \gamma(\tau) \vdash_i F$. Furthermore, for $\tau > \tau_0$, since the phase of $T[\tau_0]$ is $n \geq n_{IO}$, we deduce that F is not an attacker fact. Thus, if $\tau \in \{\tau_0 + 1; \tau_0 + 2\}$ then F is either $\mathrm{msg}_n(N[], M)$ or a table fact. But $\mathcal{T}(T[\tau_0 + 1]) = \mathcal{T}(T'[\tau])$ hence $T, \gamma(\tau) \vdash_i F$. Finally, if $\tau > \tau_0 + 2$, we know that $T'[\tau - 1] \xrightarrow{\ell}_i T'[\tau]$ implies $T[\tau - 2] \xrightarrow{\ell}_i T[\tau - 1]$ with $\mathcal{T}(T[\tau - 1]) = \mathcal{T}(T'[\tau])$. Hence, $T, \tau - 1 \vdash_i F$ and so item 1 holds. For item 3, the only possible problematic cases occur when $\gamma(\tau_1) = \gamma(\tau_2) = \tau_0 + 1$, meaning that $\tau_1, \tau_2 \in \{\tau_0 + 1; \tau_0 + 2\}$. But for theses steps, there is no event F such that $T, \tau_1 \vdash_i F$. Thus, item 3 holds.

The properties on θ are proved in a similar way: For all $(\tau, F) \in S$, if $\tau \leq \tau_0$ then we directly have that $T', \tau \vdash_{is} F$. Moreover, if $\tau > \tau_0$ then we know that $T[\tau - 1] \xrightarrow{\ell}_i T[\tau]$ implies $T'[\tau] \xrightarrow{\ell}_i T'[\tau + 1]$ with $T'[\tau + 1]$ being the same configuration as $T[\tau]$ except that $\mathcal{A}(T'[\tau + 1]) = \mathcal{A}(T[\tau]) \cup \{M'\}$. Thus, $T, \tau \vdash_{is} F$ implies $T', \tau + 1 \vdash_{is} F$ and so item 4 holds. Finally, for item 5, it suffices to notice that if $T, \tau \vdash_{is} F_1$ and $T, \tau \vdash_{is} F_2$ with F_1, F_2 two events facts then $F_1 = F_2$. Hence if $\theta(\tau_1, F_1) = \theta(\tau_2, F_2)$ then $\tau_1 = \tau_2$ by construction of θ and so $F_1 = F_2$. □

Lemma 22. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $T \in \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)$. Let $\tau_0 \in steps(T)$ such that $T[\tau_0] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1]$ by application of the rule* I-Out*. Consider the trace T' build by adding after the* I-Out *rule an instance of the* I-Msg *rule with the same label* $\mathrm{msg}(N, M)$*. Formally, T' is defined as follows:*

$$T[0] \xrightarrow{\ell_1}_i \ldots \xrightarrow{\ell_{\tau_0}}_i T[\tau_0] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1] \xrightarrow{\ell_{\tau_0+2}}_i \ldots \xrightarrow{\ell_{\tau_0+k}}_i T[\tau_0 + k]$$

where $k = \mathsf{max}_{step}(T) - \tau_0$ *and the transition* $T[\tau_0 + 1] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1]$ *is an application of the* I-Msg *rule.*

We have that $(\vdash_{is}, \{T\})$ *is mapped by* $(\vdash_{is}, \{T'\})$.

Proof. Let S be a finite set such that $S \subseteq \{(\tau, F) \in steps(T) \times \mathbb{F} \mid T, \tau \vdash_{is} F\}$. Let us define the following mapping γ and θ:

For all $\tau \in steps(T')$,

$$\gamma(\tau) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 + 1 \\ \tau - 1 & \text{if } \tau > \tau_0 + 1 \end{cases}$$

For all $(\tau, F) \in S$,

$$\theta(\tau, F) = \begin{cases} \tau & \text{if } \tau \leq \tau_0 + 1 \\ \tau + 1 & \text{if } \tau > \tau_0 + 1 \end{cases}$$

Since the trace T' contains exactly the same configurations as T with one configuration being "duplicated", that is $T[\tau_0+1]$, the proof of γ and θ satisfying the desired properties is a direct application of their definition. □

Lemma 23. *Let* $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ *be an initial instrumented configuration. Let* $n_{IO} \in \mathbb{N}$. $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \to_i))$ *is* n_{IO}*-mapped by* $(\vdash_{is}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i))$.

Proof. Thanks to Lemma 16, we only need to show how we transform a trace $T \in \mathsf{trace}(\mathcal{C}_I, \to_i)$ into a trace $T' \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$ by only using the transformations described in Lemmas 20, 21, and 22.

First, we look at the greater natural number N in T and we insert the transitions $\xrightarrow{\text{I-App}(zero)}_i \ldots \xrightarrow{\text{I-App}(succ,N-1)}_i$ at the beginning of the trace (using the transformation described in Lemma 20), yielding a trace T_1. Second, we remove in T_1 all applications of the rule I-I/O on a channel from $\mathcal{A}_I$ at a phase $n \geq n_{IO}$ by successive application of the transformation from Lemma 21, thus yielding a trace T_2. Third, after all applications of the rule I-Out in T_2, we add an application of the I-Msg with the same message and channel, i.e. the transformation described in Lemma 22, thus yielding a trace T_3.

It remains to make T_3 data-compliant. This is done by applying successively the following transformations by going through the steps in increasing order:

- When the τ-th step of T_3 is a transition $\xrightarrow{\mathrm{msg}(N,M)}_i$ by application of the rule I-Out or a transition $\xrightarrow{\text{I-App}(f,M_1,\ldots,M_m)}_i$ yielding the message M with f not being a data construction symbol or a projection of a data construction symbol, then we insert after the τ-th step the necessary transitions to deconstruct/reconstruct M, yielding a trace T_4 such that $T_4[\tau] = T_3[\tau]$ and $T_4[\tau] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T_4[\tau']$ for some τ'. Note that the added transitions are only applications of the rule I-App with data constructor function symbols or projections of a data constructor function symbol. Hence these additions correspond to the transformation described in Lemma 20.

- When the τ-th step of T_3 is a phase transition, we consider the terms in attacker knowledge $\mathcal{A}(T_3[\tau]) = \{M_1, \ldots, M_m\}$ that we order by increasing size, i.e. $i \leq j$ implies $|M_i| \leq |M_j|$. Then we insert after the τ-th step the necessary transitions to obtain a trace T_4 such that $T_4[\tau] = T_3[\tau]$ and $T_4[\tau] \overset{M_1}{\underset{\mathsf{r}}{\Longrightarrow}} \ldots \overset{M_m}{\underset{\mathsf{r}}{\Longrightarrow}} T_4[\tau']$ for some τ'. Once again,

note that the added transitions are only applications of the rule I-App with data constructor function symbols or projections of a data constructor function symbol. Hence these additions correspond to the transformation described in Lemma 20.

- When the τ-th step is any other transition, we do not modify the trace.

By construction, the yielded trace is in $\mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$ which allows us to conclude. □

A.3 $(\vdash_{is}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$ is n_{IO}-mapped by $(\vdash^{n_{IO}}_{IO}, \vdash_i, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$

Lemma 24. *Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $n_{IO} \in \mathbb{N}$. $(\vdash_{is}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$ is n_{IO}-mapped by $(\vdash^{n_{IO}}_{IO}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$.*

Proof. Let $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. Let S be a finite set such that $S \subseteq \{(\tau, F) \in \mathit{steps}(T) \times \mathbb{F} \mid T, \tau \vdash_{is} F\}$. To prove $(\vdash_{is}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$ is n_{IO}-mapped by $(\vdash^{n_{IO}}_{IO}, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i))$, we need to find a trace T' and two mappings γ and θ that satisfy the properties stated in Definition 34.

We take $T' = T$ and γ the identity function. Thus we trivially deduce that items 1, 2 and 3 of Definition 34 hold. In the following, we define θ and show at the same time that it satisfies item 4.

For all $(\tau, F) \in S$,

- if F is an event fact or a table fact then we define $\theta(\tau, F) = \tau$. By definition of $\vdash^{n_{IO}}_{IO}$, we have that $T, \tau \vdash_{is} F$ implies $T, \tau \vdash^{n_{IO}}_{IO} F$. Thus, since $(\tau, F) \in S$, we conclude that $T, \tau \vdash^{n_{IO}}_{IO} F$.

- if $F = \mathrm{msg}_i(N', M')$ then $T, \tau \vdash_{is} F$ implies $T[\tau - 1] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau]$ with $T[\tau] = i, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$.

  If I-Out is the rule applied for this transition, $i \geq n$, $N' = N[]$, $M' = M\rho$ with $N \in \mathcal{A}_I$ then we know by definition of $\mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$ that $T[\tau] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau'] \xrightarrow{\mathrm{msg}(N',M')}_i T[\tau'+1]$ for some τ' where $T[\tau'] \xrightarrow{\mathrm{msg}(N',M')}_i T[\tau' + 1]$ is an application of the rule I-Msg. In such a case, we define $\theta(\tau, F) = \tau' + 1$. Note that since $T[\tau'] \xrightarrow{\mathrm{msg}(N',M')}_i T[\tau' + 1]$ is an application of the rule I-Msg, we have that $T, \tau' + 1 \vdash^{n_{IO}}_{IO} F$ and so $T, \theta(\tau, F) \vdash^{n_{IO}}_{IO} F$

  Otherwise, we define $\theta(\tau, F) = \tau$. In such case, by definition of $\vdash^{n_{IO}}_{IO}$, we have that $T, \tau \vdash_{is} F$ implies $T, \tau \vdash^{n_{IO}}_{IO} F$ and so the result holds.

- if $F = \mathrm{att}_i(M')$ then $T, \tau \vdash_{is} F$ implies that $M' = M\rho$ and $M \in \mathcal{A}$ with $T[\tau] = i, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$. Either $M \in \mathsf{data}(T, \tau)$ and in such a case, we define $\theta(\tau, F) = \tau$ or else we know by definition of $\mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$ (specifically item 1 of Definition 12) that there exists two steps $\tau', \tau'' \in \mathit{steps}(T)$ such that $T[\tau'] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau'']$. Moreover by construction of $T[\tau'] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau'']$, we have that $M \in \mathsf{data}(T, \tau'')$. Hence we define $\theta(\tau, F) = \tau''$. Since $M \in \mathsf{data}(T, \tau'')$, we have by definition of $\vdash^{n_{IO}}_{IO}$ that $T, \tau'' \vdash^{n_{IO}}_{IO} F$ and so the result holds.

Finally, since for all $(\tau, F) \in S$, $\theta(\tau, F) = \tau$ when F is an event fact then item 5 of Definition 34 directly holds. □

We finish this section by combining all our different intermediate results to prove Lemma 8.

Lemma 8. *Let $\mathcal{C}_I = \rho, P, \mathcal{A}$ be an initial instrumented configuration. Let $\kappa \in \mathbb{N}$. Let ϱ be an IO-κ-compliant correspondence query and $\mathcal{R}$ be a set of fully IO-κ-compliant restrictions such that names$(\varrho, \mathcal{R}) \subseteq \mathrm{dom}(\rho)$.*

We have $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}) \models \varrho$ if and only if $(\vdash^{\kappa}_{IO}, \vdash_i, \mathsf{trace}^{\kappa}_{IO}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}) \models \varrho$.

Proof. The completeness of our restriction is given by Lemma 15. For the soundness, by Lemma 19, $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is mapped by $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$. Hence by Lemma 16, $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is n-mapped by $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$. By Lemma 23, $(\vdash_{is}, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is n-mapped by $(\vdash_{is}, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i))$. By Lemma 24, $(\vdash_{is}, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i))$ is n-mapped by $(\vdash^{n_{IO}}_{IO}, \vdash_i, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i))$. Using the transitivity of the mappings, i.e. Lemma 16, we obtain that $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i))$ is n-mapped by $(\vdash^{n_{IO}}_{IO}, \vdash_i, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i))$. Finally, using Corollary 1, we conclude that $(\vdash^n_{IO}, \vdash_i, \mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i)) \models \psi$ implies $(\vdash_i, \mathsf{trace}(\mathcal{C}_I, \rightarrow_i)) \models \psi$. □

B Proof of Theorem 1

Before proving Theorem 1, we state an invariant on the traces in $\mathsf{trace}^n_{IO}(\mathcal{C}_I, \rightarrow_i)$.

Definition 35. *Let $\mathcal{S}_p$ be a set of predicates. Let $n_{IO} \in \mathbb{N}$. Let $\mathcal{C}_I = \rho_0, P_0, \mathcal{A}_0$ be an initial instrumented configuration. Let $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. Let $\tau \in \{0, \ldots, \mathsf{max}_{step}(T)\}$. Let $\mathbb{C}(\tau) = \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}^{\leq\tau}_{\dot{e}}(T)$. We say $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$ holds if and only if $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Delta$ and:*

1. *for all $M \in \mathcal{A}$, there exists a derivation $\mathcal{D}'$ of $\mathrm{att}_n(M\rho)$ at step $\tau' \leq \tau$ from $\mathbb{C}(\tau')$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and τ' is the smallest step such that $T, \tau' \vdash^{n_{IO}}_{IO} \mathrm{att}_n(M\rho)$; and*
2. *for all $tbl(M_1, \ldots, M_r) \in \mathcal{T}$, there exists a derivation $\mathcal{D}'$ of $\mathrm{table}_n(tbl(M_1, \ldots, M_r)\rho)$ at step $\tau' \leq \tau$ from $\mathbb{C}(\tau)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and τ' is the smallest step such that $T, \tau' \vdash^{n_{IO}}_{IO} \mathrm{table}_n(tbl(M_1, \ldots, M_r)\rho)$; and*

for all $(P, \mathcal{O}, \mathcal{I}) \in \mathcal{P}$, there exist $P', \mathcal{H}', \mathcal{I}', \rho', \sigma'$ such that:

3. *$[\![P', \mathcal{O}, \mathcal{I}']\!] n \mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$; and*
4. *$P\rho = P'\rho'\sigma'$, $\mathcal{I}'\sigma' = \mathcal{I}$; and*
5. *for all $F' \in \mathcal{H}'\sigma'$, there exist τ' and a derivation $\mathcal{D}'$ of F' at step τ' from $\mathbb{C}(\tau')$ such that $\tau' \leq \tau$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. For all formulae $\phi \in \mathcal{H}'$, $\sigma' \models \phi$.*

Items 1 and 2 indicate that any terms added to the attacker knowledge or in a table can be derived by the set of initial clauses. Note that the trace T always satisfies these derivations w.r.t. the set of allowed predicates $\mathcal{S}_p$. Items 3 and 4 show the link between the concrete trace and the symbolic representation of the process within the transformation $[\![P, \mathcal{O}, \mathcal{I}]\!] n \mathcal{H}\rho$. Similarly to the first two items, item 5 also indicates that the concretisation of facts in $\mathcal{H}$ can be derived by the set of initial clauses.

In Definition 35, we define an invariant on trace that we will rely on to prove Lemma 1. Contrary to what could be expected, given a trace T, we will *not* show that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$ for all steps $\tau \in \{0, \ldots, \mathsf{max}_{step}(T)\}$. The main reason being that all derivations in Definition 35 must be satisfied by the trace w.r.t. the set of allowed predicates $\mathcal{S}_p$. Such a property is in fact

problematic for terms that have a data constructor function symbol as root. However, since T is data compliant, we know from Definition 12 that the trace always deconstructs/reconstructs terms when they are first introduced. We will in fact show that our invariant always holds at the end of such deconstruction/reconstruction phase.

B.1 Handling data constructor function symbols

Given a trace T and a step τ, if $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Delta$, we denote by $\mathcal{A}(T[\tau])$ the set $\mathcal{A}$.

Given a derivation $\mathcal{D}$, a trace T and a set of allowed predicates $\mathcal{S}_p$, we consider a weaker notion of satisfiability, denoted $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$, that follows Definition 18 except that item 4 of Definition 18 is weakened in the case where $F_0 = \mathrm{att}_i(f(M_1, \ldots, M_n))$ with $f \in \mathcal{F}_{data}$ when η is the root of the derivation. In such case, we only request that $f(M_1, \ldots, M_n) \in \mathcal{A}(T[\tau_0])$.

Finally, we denote by $\mathcal{A}_d(T[\tau])$ and $\mathcal{A}_w(T[\tau])$ the two disjoint sets such that $\mathcal{A}_w(T[\tau])$ is the smallest set satisfying the following properties:

- $\mathcal{A}(T[\tau]) = \mathcal{A}_d(T[\tau]) \cup \mathcal{A}_w(T[\tau])$
- for all $M \in \mathcal{A}_w(T[\tau])$, there exist $\tau' \leq \tau$ and a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step τ' from $\mathbb{C}(\tau')$ such that $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$; and
- for all $M \in \mathcal{A}_d(T[\tau])$, there exists a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step $\tau' \leq \tau$ from $\mathbb{C}(\tau')$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and τ' is the smallest step such that $T, \tau' \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$.
- for all $M \in \mathcal{A}(T[\tau])$, $M \in \mathcal{A}_d(T[\tau])$ if and only if there exists $\tau' \leq \tau$ such that $T, \tau' \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$.

Note that $\mathcal{A}_d(T[\tau])$ and $\mathcal{A}_w(T[\tau])$ do not necessarily exist (for instance if $M \in \mathcal{A}(T[\tau])$ and there is no derivation of $\mathrm{att}_n(M\rho)$).

Lemma 25. *Let $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$. Let $\tau \in \mathbb{N}$. Assume that $\mathcal{A}_d(T[\tau])$ and $\mathcal{A}_w(T[\tau])$ exist. For all $M \in \mathcal{A}(T[\tau])$, if $T[\tau] \overset{M}{\underset{\mathsf{r}}{\Longrightarrow}} T[\tau']$ or $T[\tau] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau']$, then $\mathcal{A}_d(T[\tau'])$ and $\mathcal{A}_w(T[\tau'])$ exist with $\mathcal{A}_w(T[\tau']) = \mathcal{A}_w(T[\tau]) \setminus \{M\}$ and $M \in \mathcal{A}_d(T[\tau'])$.*

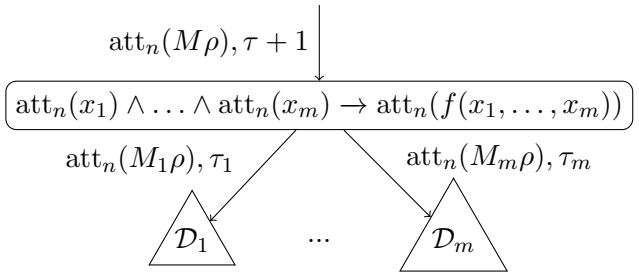
Proof. We prove this lemma by induction on the size of M. Consider $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Delta$. Since M is a ground term, we have $M = f(M_1, \ldots, M_m)$.

If $M \in \mathcal{A}_d(T[\tau])$, then by definition, there exists $\tau'' \leq \tau$ such that $T, \tau'' \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$. Hence $M \in \mathsf{data}(T, \tau'')$ which implies $M \in \mathsf{data}(T, \tau)$. By definition of $T[\tau] \overset{M}{\underset{\mathsf{r}}{\Longrightarrow}} T[\tau']$ and $T[\tau] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau']$, we deduce in both cases that $\tau' = \tau$. As such, $\mathcal{A}_d(T[\tau'])$ and $\mathcal{A}_w(T[\tau'])$ trivially exist. Moreover, since $\mathcal{A}_d(T[\tau'])$ and $\mathcal{A}_w(T[\tau'])$ are disjoint and $M \in \mathcal{A}_d(T[\tau])$, we obtain that $\mathcal{A}_w(T[\tau']) \setminus \{M\} = \mathcal{A}_w(T[\tau']) = \mathcal{A}_w(T[\tau])$.

If $M \notin \mathcal{A}_d(T[\tau])$, i.e. $M \in \mathcal{A}_w(T[\tau])$, then $f \in \mathcal{F}_{data}$ (otherwise $T, \tau \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$, which would imply $M \in \mathcal{A}_d(T[\tau])$).

We focus first on $T[\tau] \overset{M}{\underset{\mathsf{r}}{\Longrightarrow}} T[\tau']$: By definition, $\tau' = \tau + 1$ and $T[\tau] \xrightarrow{\text{I-App}(f, M_1, \ldots, M_m)}_i T[\tau + 1]$ and $M_1, \ldots, M_m \in \mathsf{data}(T, \tau)$. Thus, for all $i \in \{1, \ldots, m\}$, $T, \tau \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M_i\rho)$ and so $M_1, \ldots, M_m \in \mathcal{A}_d(T[\tau])$. Hence, for all $i \in \{1, \ldots, m\}$, there exists a derivation $\mathcal{D}_i$ of $\mathrm{att}_n(M_i\rho)$ at step $\tau_i \leq \tau$ from $\mathbb{C}(\tau_i)$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ where τ_i is the smallest step such that

$T, \tau_i \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M_i\rho)$. By definition, we know that $f(x_1, \ldots, x_m) \to f(x_1, \ldots, x_m) \in \mathsf{def}(f)$. Hence we can thus build a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step $\tau + 1$ by applying the rule Rf on $\mathcal{D}_1, \ldots, \mathcal{D}_m$:

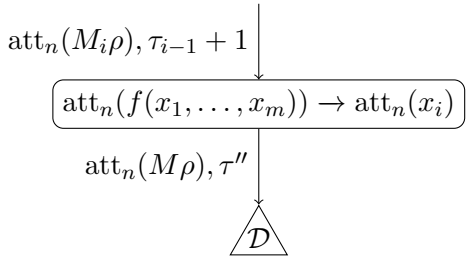


Since $T[\tau] \xrightarrow{\text{I-App}(f,M_1,\ldots,M_m)}_i T[\tau + 1]$ and $f(M_1, \ldots, M_m) \in \mathcal{A}(T[\tau + 1])$, we deduce that $T, \tau + 1 \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$ and so $\mathcal{D}$ satisfies item 4 of Definition 18. Moreover, since $\tau_i \leq \tau$ for all i, we also deduce that item 3 of Definition 18 is satisfied. Items 1 and 2 are trivially satisfied since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ for all i. Therefore, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. Note that since $M \notin \mathcal{A}_d(T[\tau])$, we do have that $\tau + 1$ is the smallest step such that $T, \tau + 1 \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M\rho)$ meaning that $M \in \mathcal{A}_d(T[\tau + 1])$ and so $M \notin \mathcal{A}_w(T[\tau + 1])$. Since the attacker knowledge of $T[\tau]$ and $T[\tau + 1]$ is the same (M was already in $\mathcal{A}$), we can conclude.

We now focus on $T[\tau] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau']$. By definition, there exist $\tau_0, \ldots, \tau_m$ such that $\tau' = \tau_m + 1$, $\tau_0 = \tau$ and $T[\tau_0] \overset{M}{\underset{\mathsf{d},1}{\Longrightarrow}} T[\tau_1] \overset{M}{\underset{\mathsf{d},2}{\Longrightarrow}} \ldots \overset{M}{\underset{\mathsf{d},m}{\Longrightarrow}} T[\tau_m] \overset{M}{\underset{\mathsf{r}}{\Longrightarrow}} T[\tau_m + 1]$. Let us show that for all $i \leq m$, $\{M_1, \ldots, M_i\} \subseteq \mathcal{A}_d(T[\tau_i])$ and $\mathcal{A}_w(T[\tau_i]) = \mathcal{A}_w(T[\tau])$. This proof is done by induction on i. The base case ($i = 0$) being trivial, we only focus on the inductive step. By our inductive hypothesis, we know that $\mathcal{A}_d(T[\tau_{i-1}])$ and $\mathcal{A}_w(T[\tau_{i-1}])$ exist, $\mathcal{A}_w(T[\tau_{i-1}]) = \mathcal{A}_w(T[\tau])$ and $\{M_1, \ldots, M_{i-1}\} \subseteq \mathcal{A}_d(T[\tau_{i-1}])$. We do a case analysis on $T[\tau_{i-1}] \overset{M}{\underset{\mathsf{d},i}{\Longrightarrow}} T[\tau_i]$:

- *Case $\tau_i = \tau_{i-1}$ and $M_i \in \mathsf{data}(T, \tau_{i-1})$:* In such a case, $T, \tau_i \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M_i\rho)$ and so $M_i \in \mathcal{A}_d(T[\tau_{i-1}])$. Since our inductive hypothesis gave us $\mathcal{A}_w(T[\tau_{i-1}]) = \mathcal{A}_w(T[\tau])$ and $\{M_1, \ldots, M_{i-1}\} \subseteq \mathcal{A}_d(T[\tau_{i-1}])$. We conclude that $\mathcal{A}_w(T[\tau_i]) = \mathcal{A}_w(T[\tau])$ and $\{M_1, \ldots, M_i\} \subseteq \mathcal{A}_d(T[\tau_i])$.

- *Case $T[\tau_{i-1}] \xrightarrow{\text{I-App}(\pi_i^f,M)}_i T[\tau_{i-1} + 1] \overset{M_i}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau_i]$:* Since $M \in \mathcal{A}_w(T[\tau])$, we know that there exists τ'' and a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step τ'' from $\mathbb{C}(\tau'')$ such that $\tau'' \leq \tau_{i-1}$ and $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$. Since $T[\tau_{i-1}] \xrightarrow{\text{I-App}(\pi_i^f,M)}_i T[\tau_{i-1} + 1]$, we can build a derivation $\mathcal{D}_i$ of $\mathrm{att}_n(M_i\rho)$ at step $\tau_{i-1} + 1$ by applying the rule related the projection π_i^f:



Thanks to $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$ and the root node of the derivation $\mathcal{D}_i$ being the application of a projection of f, we obtain $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}_i$. Hence, $\mathcal{A}_w(T[\tau_{i-1}+1])$ and $\mathcal{A}_d(T[\tau_{i-1}+1])$ exist.

If the root of M_i is not a data constructor symbol, then $T, \tau_{i-1}+1 \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M_i\rho)$ and so $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$. Note that we know that $M_i \notin \mathsf{data}(T, \tau_{i-1})$ meaning that $\tau_{i-1}+1$ is the smallest step such that $T, \tau_{i-1}+1 \vdash_{IO}^{n_{IO}} \mathrm{att}_n(M_i\rho)$. Hence, $M_i \in \mathcal{A}_d(T[\tau_{i-1}+1])$ and $\mathcal{A}_w(T[\tau_{i-1}+1]) = \mathcal{A}_w(T[\tau_{i-1}])$. Moreover, by definition of $T[\tau_{i-1}+1] \xRightarrow[\mathsf{dr}]{M_i} T[\tau_i]$, $\tau_i = \tau_{i-1}+1$. Therefore we can conclude that $\mathcal{A}_w(T[\tau_i]) = \mathcal{A}_w(T[\tau])$ and $\{M_1, \ldots, M_i\} \subseteq \mathcal{A}_d(T[\tau_i])$.

If the root of M_i is a data constructor symbol, then $\mathcal{A}_w(T[\tau_{i-1}+1]) = \mathcal{A}_w(T[\tau_{i-1}]) \cup \{M_i\}$. Indeed, we cannot have $M_i \in \mathsf{data}(T, \tau_{i-1}+1)$ since $M_i \notin \mathsf{data}(T, \tau_{i-1})$ and $T[\tau_{i-1}] \xrightarrow{\text{I-App}(\pi_i^f, M)}_i T[\tau_{i-1}+1] \xRightarrow[\mathsf{dr}]{M_i} T[\tau_i]$. Since $T[\tau_{i-1}+1] \xRightarrow[\mathsf{dr}]{M_i} T[\tau_i]$ and $|M_i| < |M|$, we can apply our main inductive hypothesis which allows us to deduce that $\mathcal{A}_w(T[\tau_i]) = \mathcal{A}_w(T[\tau_{i-1}+1])\backslash\{M_i\}$ and $M_i \in \mathcal{A}_d(T[\tau_i])$. Thus, $\mathcal{A}_w([\tau_i]) = \mathcal{A}_w(T[\tau_{i-1}]) = \mathcal{A}_w(T[\tau])$. Moreover, since $\{M_1, \ldots, M_{i-1}\} \subseteq \mathcal{A}_d(T[\tau_{i-1}])$, we conclude that $\{M_1, \ldots, M_i\} \subseteq \mathcal{A}_d(T[\tau_i])$.

We have shown that $\{M_1, \ldots, M_m\} \subseteq \mathcal{A}_d(T[\tau_m])$ and $\mathcal{A}_w(T[\tau_m]) = \mathcal{A}_w(T[\tau])$. Since $T[\tau_m] \xRightarrow[\mathsf{r}]{M} T[\tau']$ with $M = f(M_1, \ldots, M_m)$, we have already shown that $\mathcal{A}_d(T[\tau'])$ and $\mathcal{A}_w(T[\tau'])$ exist and $\mathcal{A}_w(T[\tau']) = \mathcal{A}_w(T[\tau_m])\backslash\{M\}$. With $\mathcal{A}_w(T[\tau_m]) = \mathcal{A}_w(T[\tau])$, we conclude our proof. □

B.2 Proving the invariant

We start by showing the soundness of $D' \Downarrow' (U', \sigma', \phi)$ with respect to $D \Downarrow U$. The following lemma is the same as [BAF08, Lemma 11] but extended to may-fail terms and rewrite rules with conditional formulae.

Lemma 26 ([CB13])**.** *Let σ be a closed substitution.*

Let D be a plain expression. If $D\sigma \Downarrow U$, then there exist U', σ_1, ϕ and σ_1' such that $D \Downarrow' (U', \sigma_1, \phi)$, $U = U'\sigma_1'$, $\sigma = (\sigma_1\sigma_1')_{|dom(\sigma)}$ and $\sigma_1' \models \phi$.

Let $D_1, \ldots, D_n$ be plain expressions. If for all $i \in \{1, \ldots, n\}$, $D_i\sigma \Downarrow U_i$, then there exist $U_1', \ldots, U_n'$, σ_1, ϕ and σ_1' such that $(D_1, \ldots, D_n) \Downarrow' ((U_1', \ldots, U_n'), \sigma_1, \phi)$, $U_i = U_i'\sigma_1'$ for all $i \in \{1, \ldots, n\}$, $\sigma = (\sigma_1\sigma_1')_{|dom(\sigma)}$ and $\sigma_1' \models \phi$.

Lemma 1 considers IO-compliant traces. From Definition 12, a trace $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i)$ has a particular shape:

- The first transitions of T are the transitions $\xrightarrow{\text{I-App}(0)}_i \xrightarrow{\text{I-App}(succ,0)}_i \cdots \xrightarrow{\text{I-App}(succ,n-1)}_i$ for some integer n.
- An output transition is followed by a deconstruction/reconstruction phase when needed, i.e. if $T[\tau] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau']$ by the rule I-Out then $T[\tau'] \xRightarrow[\mathsf{dr}]{M} T[\tau'']$ for some τ''. Moreover, if $n \geq n_{IO}$ and $N \in \mathcal{A}_I$ then $T[\tau''] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau''+1]$ by the rule I-Msg.

- An function application transition is also followed by a deconstruction/reconstruction phase when needed, i.e. if $T[\tau] \xrightarrow{\text{I-App}(f,M_1,\ldots,M_n)}_i T[\tau']$ with $f(M_1,\ldots,M_n) \Downarrow M$ then $T[\tau'] \overset{M}{\Longrightarrow}_{\mathsf{dr}} T[\tau'']$ for some τ''.
- A phase transition is followed by a reconstruction of all terms in the attacker knowledge, i.e. if $T[\tau_0 - 1] \to_i T[\tau_0]$ by the rule I-Phase then $T[\tau_0] \overset{M_1}{\Longrightarrow}_{\mathsf{r}} T[\tau_1] \overset{M_2}{\Longrightarrow}_{\mathsf{r}} \ldots \overset{M_n}{\Longrightarrow}_{\mathsf{r}} T[\tau_n]$ for some $\tau_1,\ldots,\tau_n$ and $\mathcal{A}(T[\tau_0]) = \{M_1,\ldots,M_n\}$.

We call these sequences of transitions *complete transitions*, denoted $\to_c$. We also define $\mathcal{C} \to_c \mathcal{C}'$ when $\mathcal{C} \to_i \mathcal{C}'$ is an application of a rule other than I-Out, I-App and I-Phase. Therefore, a trace T can be seen as a sequence of complete transitions, i.e. $T = T[0] \to_c T[\tau_1] \to_c \ldots \to_c T[\tau_n]$ where $\tau_1,\ldots,\tau_{n-1} \leq \mathsf{max}_{step}(T)$ and $\tau_n = \mathsf{max}_{step}(T)$. We say that $0,\tau_1,\ldots,\tau_n$ are the *complete steps* of T. In the next lemma we show that $\mathcal{I}nv_{\mathcal{S}_p}(T,\tau)$ for any complete steps τ of T.

Lemma 27. *Let $\mathcal{S}_p$ be a set of predicates. Let $n_{IO} \in \mathbb{N}$. Let $\mathcal{C}_I = \rho_I, P_I, \mathcal{A}_I$ be an initial instrumented configuration. For all $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$, for all complete steps τ of T, $\mathcal{I}nv_{\mathcal{S}_p}(T,\tau)$.*

Proof. First of all, note that for all derivations $\mathcal{D}'$ and all sets of predicates $\mathcal{S}_p, \mathcal{S}'_p$, Definition 18 directly gives us that $\mathcal{S}_p \subseteq \mathcal{S}'_p$ and $T, \mathcal{S}'_p, n_{IO} \vdash \mathcal{D}'$ implies $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. Therefore, $\mathcal{S}_p \subseteq \mathcal{S}'_p$ and $\mathcal{I}nv_{\mathcal{S}'_p}(T,\tau)$ implies $\mathcal{I}nv_{\mathcal{S}_p}(T,\tau)$. Thus, it is sufficient to prove the lemma when $\mathcal{S}_p$ contains all predicates. The proof is done by induction on τ.

Base case $\tau = 0$: $T[0] = \mathcal{C}_I = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ with $\mathcal{P} = \{\!\{(P_I, \emptyset, \emptyset)\}\!\}$, $\rho = \rho_I$, $\mathcal{A} = \mathcal{A}_I$, $\mathcal{T} = \emptyset$, $\Lambda = \emptyset$ and $n = 0$. By defining $P' = P_I, \mathcal{I}' = \emptyset$, $\mathcal{H}' = \top$, $\rho' = \rho_I$ and $\sigma' = id$, we directly obtain that item 4 of Definition 35 holds. Moreover, by definition, $\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) = [\![(P_I, \emptyset, \emptyset)]\!] 0 \top \rho_I$ hence the item 3 holds. Since $\mathcal{H}'\sigma' = \top$ and $\mathcal{T} = \emptyset$, items 2 and 5 also hold. By definition of an initial instrumented semantics, $\mathcal{A}_I$ only contains names that are included in $dom(\rho_I)$. Moreover, for all $a \in \mathcal{A}_I$, $a\rho_I = a[]$. Hence for all $a \in \mathcal{A}_I$, we can build a derivation of $\mathrm{att}_0(a[])$ that is satisfied by T at step 0 w.r.t. $\mathcal{S}_F$ by using the clauses RInit from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I, n_{IO})$. Therefore, we conclude that $\mathcal{I}nv_{\mathcal{S}_p}(T,0)$ holds.

Inductive step $\tau > 0$: Since $\tau > 0$ and τ is a complete step, we know that there exists a complete step $\tau_0 < \tau$ such that $T[\tau_0] \to_c T[\tau]$. By our inductive hypothesis, we deduce that $\mathcal{I}nv_{\mathcal{S}_p}(T,\tau_0)$ holds. Note that when the transition $T[\tau_0] \to_c T[\tau]$ does not correspond to a change of phase, it is sufficient to prove the invariant on the modified elements between $T[\tau_0]$ and $T[\tau]$ to prove $\mathcal{I}nv_{\mathcal{S}_p}(T,\tau)$ (the invariant directly holds on unmodified elements in $T[\tau]$ from the fact the invariant holds on $T[\tau_0]$). Let us denote $T[\tau_0] = n_0, \rho_0, \mathcal{P}_0, \mathcal{T}_0, \mathcal{A}_0, \Lambda_0$ and $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$.

We do a case analysis on the complete transition $T[\tau_0] \to_c T[\tau]$:

- Rule I-Nil: Trivial since $\mathcal{P} \subset \mathcal{P}_0$.
- Rule I-Msg: Trivial since the derivations are unchanged.
- Rule I-Par: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(P \mid Q, \mathcal{O}, \mathcal{I})\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P, \mathcal{O}, \mathcal{I}), (Q, \mathcal{O}, \mathcal{I})\}\!\}$, $\rho = \rho_0, n = n_0$. By our inductive hypothesis, we know that there exist $P', Q', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(P' \mid Q')\rho'\sigma' = (P \mid Q)\rho$, $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![P' \mid Q', \mathcal{O}, \mathcal{I}']\!] n \mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

By definition, it implies that $[\![P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$ and $[\![Q', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. Thus the result holds by associating $P', \mathcal{H}', \mathcal{I}', \rho', \sigma'$ (resp. $Q', \mathcal{H}', \mathcal{I}', \rho', \sigma'$) to $(P, \mathcal{O}, \mathcal{I})$ (resp. $(Q, \mathcal{O}, \mathcal{I})$).

- Rule I-Repl: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(!^o P, \mathcal{O}, \mathcal{I})\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P, (\mathcal{O}, o), (\mathcal{I}, \lambda))\}\!\}$, $\rho = \rho_0$, $n = n_0$, $\Lambda = \Lambda_0 \cup \{\lambda\}$ with $\lambda \notin \Lambda_0$. By our inductive hypothesis, we know that there exist $!^o P', \mathcal{H}', \mathcal{I}', \rho', \sigma'$ such that $[\![!^o P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$, $(!^o P)\rho = (!^o P')\rho'\sigma'$ and $\mathcal{I}'\sigma' = \mathcal{I}$. By definition $[\![!^o P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' = [\![P', (\mathcal{O}, o), (\mathcal{I}', i)]\!] n\mathcal{H}'\rho'$. Let us define $\sigma'' = \sigma'[i \mapsto \lambda]$. We obtain that $(\mathcal{I}', i)\sigma'' = (\mathcal{I}, \lambda)$. Moreover, since i is fresh, we deduce that $P\rho = P'\rho'\sigma''$. Similarly, $\mathcal{H}'\sigma' = \mathcal{H}'\sigma''$ meaning that item 5 holds for $\mathcal{H}'\sigma''$. Hence we conclude the proof by associating $P', \mathcal{H}', (\mathcal{I}', i), \rho', \sigma''$ to $(P, (\mathcal{O}, o), (\mathcal{I}, \lambda))$.

- Rule I-Restr: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{new}\ a; P, \mathcal{O}, \mathcal{I})\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P\{^{a'}/_a\}, \mathcal{O}, \mathcal{I})\}\!\}$ and $\rho = \rho_0[a' \mapsto a[\mathcal{I}]]$ where $a' \notin dom(\rho_0)$. By our inductive hypothesis, we know that there exist $P', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{new}\ a; P')\rho'\sigma' = (\mathsf{new}\ a; P)\rho_0$ (note that since we consider application of mapping modulo renaming of bound name, we can have the same a in both $\mathsf{new}\ a; P'$ and $\mathsf{new}\ a; P$), $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{new}\ a; P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. By definition $[\![\mathsf{new}\ a; P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' = [\![P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'(\rho'[a \mapsto a[\mathcal{I}']])$.

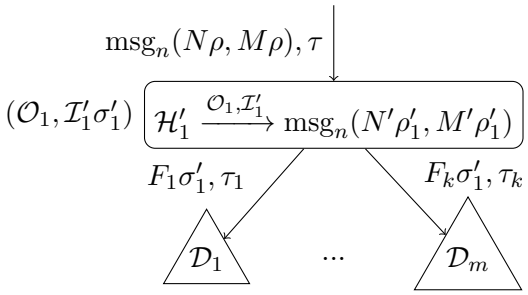
  Let us define $\rho'' = \rho'[a \mapsto a[\mathcal{I}']]$. Since $\mathcal{I}'\sigma' = \mathcal{I}$ and $a' \notin dom(\rho_0)$, we have $P\{^{a'}/_a\}\rho = P\{^{a'}/_a\}\rho_0[a' \mapsto a[\mathcal{I}]] = P\rho_0[a \mapsto a[\mathcal{I}]] = P\rho_0[a \mapsto a[\mathcal{I}']\sigma']$. With $P\rho_0 = P'\rho'\sigma'$, we obtain $P\{^{a'}/_a\}\rho = P'\rho''\sigma'$. We conclude by associating $P', \mathcal{H}', \mathcal{I}', \rho'', \sigma'$ to $(P\{^{a'}/_a\}, \mathcal{O}, \mathcal{I})$.

- Rule I-I/O: In such a case, $\tau = \tau_0 + 1$, $\rho = \rho_0$, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}_1, \mathcal{I}_1), (\mathsf{in}^o(N, x); Q, \mathcal{O}_2, \mathcal{I}_2)\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P, \mathcal{O}_1, \mathcal{I}_1), (Q\{^M/_x\}, (\mathcal{O}_2, o), (\mathcal{I}_2, M\rho))\}\!\}$. Moreover, since $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$, we have $n < n_{IO}$ or $N \notin \mathcal{A}_I$. By our inductive hypothesis, we know that there exist $N', M', P', Q', \mathcal{I}'_1, \mathcal{I}'_2, \rho'_1, \rho'_2, \mathcal{H}'_1, \mathcal{H}'_2, \sigma'_1, \sigma'_2$ such that $(\mathsf{out}(N, M); P)\rho = (\mathsf{out}(N', M'); P')\rho'_1\sigma'_1$, $(\mathsf{in}^o(N, x); Q)\rho = (\mathsf{in}^o(N'', x); Q')\rho'_2\sigma'_2$, $\mathcal{I}'_1\sigma'_1 = \mathcal{I}_1$ and $\mathcal{I}'_2\sigma'_2 = \mathcal{I}_2$. Moreover, we have $[\![\mathsf{out}(N', M'); P', \mathcal{O}_1, \mathcal{I}'_1]\!] n\mathcal{H}'_1\rho'_1 \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$ and $[\![\mathsf{in}^o(N'', x); Q, \mathcal{O}_2, \mathcal{I}'_2]\!] n\mathcal{H}'_2\rho'_2 \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$.

  By definition, since $n < n_{IO}$ or $N \notin \mathcal{A}_I$ (which implies $N' \notin \mathcal{A}_I$ and $N'' \notin \mathcal{A}_I$), we have $[\![\mathsf{out}(N', M'); P', \mathcal{O}_1, \mathcal{I}'_1]\!] n\mathcal{H}'_1\rho'_1 = [\![P', \mathcal{O}_1, \mathcal{I}'_1]\!] n\mathcal{H}'_1\rho'_1 \cup \{\mathcal{H}'_1 \xrightarrow{\mathcal{O}_1, \mathcal{I}'_1} \mathrm{msg}_n(N'\rho'_1, M'\rho'_1)\}$ and $[\![\mathsf{in}^o(N'', x); Q, \mathcal{O}_2, \mathcal{I}'_2]\!] n\mathcal{H}'_2\rho'_2 = [\![Q, (\mathcal{O}_2, o), (\mathcal{I}'_2, x')]\!] n(\mathcal{H}'_2 \wedge \mathrm{msg}_n(N''\rho'_2, x'))(\rho'_2[x \mapsto x'])$.

  Let us define $\rho''_2 = \rho'_2[x \mapsto x']$ and $\sigma''_2 = \sigma'_2[x' \mapsto M\rho]$. Since x' is fresh and x is bound, both x and x' does not occur in ρ, ρ'_2 and σ'_2. Thus, $Q\{^M/_x\}\rho = Q\rho\{^{M\rho}/_x\} = Q'\rho'_2\sigma'_2\{^{M\rho}/_x\} = Q'\rho''_2\sigma''_2$. Note that we also directly have that $(\mathcal{I}'_2, x')\sigma''_2 = (\mathcal{I}_2, M\rho)$. Therefore, items 4 and 3 hold by associating $P', \mathcal{H}'_1, \mathcal{I}'_1, \rho'_1, \sigma'_1$ to $(P, \mathcal{O}_1, \mathcal{I}_1)$ and by associating $Q', (\mathcal{H}'_2 \wedge \mathrm{msg}_n(N''\rho'_2, x')), (\mathcal{I}'_2, x'), \rho''_2, \sigma''_2$ to $(Q\{M/_x\}, (\mathcal{O}_2, o), (\mathcal{I}_2, M\rho))$. We now show the other items of Definition 35. Items 1 and 2 trivially hold since $\mathcal{A}_0 = \mathcal{A}$ and $\mathcal{T}_0 = \mathcal{T}$. Thus it remains to prove item 5 for $\mathrm{msg}_n(N''\rho'_2, x')\sigma''_2$, that is, $\mathrm{msg}_n(N''\rho'_2\sigma'_2, x'\sigma''_2) = \mathrm{msg}_n(N\rho, M\rho)$.

  Let us assume that $\mathcal{H}'_1 = F_1 \wedge \ldots \wedge F_k \wedge \phi$. Since $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds, $\sigma'_1 \models \phi$ and there exist $\tau_1, \ldots, \tau_k$ and some derivation $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $F_1\sigma'_1, \ldots, F_k\sigma'_1$ at step $\tau_1, \ldots, \tau_k$ respectively from $\mathbb{C}(\tau_0)$ such that $\tau_j \leq \tau_0$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. But the rule $\mathcal{H}'_1 \xrightarrow{\mathcal{O}_1, \mathcal{I}'_1} \mathrm{msg}_n(N'\rho'_1, M'\rho'_1)$ is in $\mathbb{C}(\tau)$ and $\tau_0 < \tau$. Therefore, we can build the following derivation $\mathcal{D}$ of $\mathrm{msg}_n(N\rho, M\rho)$ at step τ:



Since $T, \tau, n_{IO} \vdash_{is} \mathrm{msg}_n(N\rho, M\rho)$, $\tau_1 < \tau$, ..., $\tau_k < \tau$ and $(P, \mathcal{O}_1, \mathcal{I}'_1\sigma'_1) \in \mathcal{P}$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ which allows us to conclude.

- Rule I-Let1: In such a case, $\rho = \rho_0$, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P\{^M/_x\}, \mathcal{O}, \mathcal{I})\}\!\}$ with $D \Downarrow M$. By our inductive hypothesis, we know that there exist $D', P', Q', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{let}\ x = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q')\rho'\sigma' = (\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q)\rho$, $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{let}\ x = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. Since ρ is a mapping from names to name patterns such that for all $a, b \in dom(\rho)$, $a = b$ iff $a\rho = b\rho$ (thanks to Lemma 2), $D \Downarrow M$ implies $D\rho \Downarrow M\rho$. Since $D\rho = (D'\rho')\sigma'$, we can apply Lemma 26 to obtain that there exist M', σ_1, σ'_1, and ϕ such that $D'\rho' \Downarrow' (M', \sigma_1, \phi)$, $M\rho = M'\sigma'_1$, $\sigma' = (\sigma_1\sigma'_1)_{|dom(\sigma')}$ and $\sigma'_1 \models \phi$.

  By definition of $[\![\mathsf{let}\ x = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho'$, we have $[\![P', \mathcal{O}, \mathcal{I}'\sigma_1]\!]n(\mathcal{H}'\sigma_1 \wedge \phi)(\rho'\sigma_1[x \mapsto M']) \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. Let us define $P'' = P'$, $\mathcal{I}'' = \mathcal{I}'\sigma_1$, $\mathcal{H}'' = \mathcal{H}'\sigma_1 \wedge \phi$, $\rho'' = \rho'\sigma_1[x \mapsto M']$ and $\sigma'' = \sigma'_1$. We show that the invariant holds by associating $P'', \mathcal{H}'', \mathcal{I}'', \rho''$ and σ'' to $(P\{^M/_x\}, \mathcal{O}, \mathcal{I})$. We already proved that item 3 holds. Moreover, $P\{^M/_x\}\rho = (P\rho)\{^{M\rho}/_x\} = P'\rho'\sigma'\{^{M'\sigma'_1}/_x\}$. But $\sigma' = (\sigma_1\sigma'_1)_{|dom(\sigma')}$ hence $P\{^M/_x\}\rho = (P'\rho'\sigma_1\{^{M'}/_x\})\sigma'_1$ (note that $x \notin dom(\sigma'_1)$) which implies $P\{^M/_x\}\rho = P''\rho''\sigma''$. Furthermore, $\mathcal{I} = \mathcal{I}'\sigma' = \mathcal{I}'\sigma_1\sigma'_1 = \mathcal{I}''\sigma'_1$. Hence item 4 holds. Finally, the facts in $\mathcal{H}'\sigma'$ are the same as in $\mathcal{H}'\sigma_1\sigma'_1$ and so in $\mathcal{H}''\sigma''$. Since $\sigma'_1 \models \phi$, we conclude that item 5 holds.

- Rule I-Let2: Similar to the previous case.

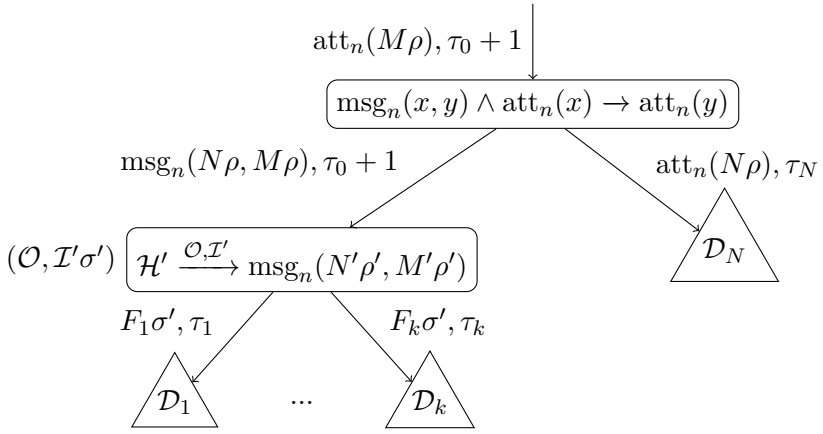
- Output transition: Since $T[\tau_0] \to_c T[\tau]$ is a complete transition, one of the following two cases hold:

  1. $T[\tau_0] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau']$, $\tau = \tau'$ and either $N \notin \mathcal{A}_I$ or $n < n_{IO}$.
  2. $T[\tau_0] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau_0 + 1] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau'] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau]$ with $\tau = \tau' + 1$, $N \in \mathcal{A}_I$, $n \geq n_{IO}$ and $T[\tau - 1] \xrightarrow{\mathrm{msg}(N,M)}_i T[\tau]$ is an application of I-Msg.

  Note that $T[\tau_0 + 1] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau']$ only modifies the attacker knowledge and the rule I-Msg leaves the configuration unchanged. Let us denote $\mathcal{A}_1$ the attacker knowledge of $T[\tau_0+1]$. We have $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}, \mathcal{I})\}\!\}$, $N \in \mathcal{A}_0$, $\mathcal{A}_1 = \mathcal{A}_0 \cup \{M\}$ and $\mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$. By our inductive hypothesis, we know that there exists $N', M', P', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{out}(N', M'); P')\rho'\sigma' = (\mathsf{out}(N, M); P)\rho$, $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{out}(N', M'); P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$.

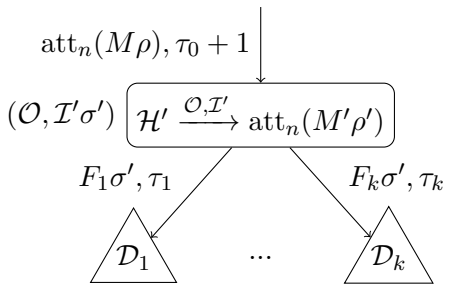
  We do a case analysis on the two previous cases:

– Case 1: by definition $[\![\mathsf{out}(N', M'); P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' = [\![P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \cup \{\mathcal{H}' \xrightarrow{\mathcal{O},\mathcal{I}'} \mathrm{msg}_n(N'\rho', M'\rho')\}$. But we know that $N \in \mathcal{A}_0$. Thus $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau')$ ensures that there exists a derivation $\mathcal{D}_N$ of $N\rho$ at some step $\tau_N \leq \tau_0$ from $\mathbb{C}(\tau_N)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_n$. Moreover, item 5 also ensures that if we denote $\mathcal{H}' = F_1 \wedge \ldots \wedge F_k \wedge \phi$ then $\sigma' \models \phi$ and there exist $\tau_1, \ldots, \tau_k$ and some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $F_1\sigma', \ldots, F_k\sigma'$ at step $\tau_1, \ldots, \tau_k$ respectively from $\mathbb{C}(\tau_0)$ such that $\tau_j \leq \tau_0$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. Thus we can build a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step $\tau_0 + 1$ from $\mathbb{C}(\tau_0 + 1)$ as follows:



Note that we do not necessarily have that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ when $M\rho$ has a data constructor function symbol at the root. However, we do have that $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$. Furthermore, since $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds and in particular item 1, we deduce that $\mathcal{A}_d(T[\tau_0 + 1])$ and $\mathcal{A}_w(T[\tau_0 + 1])$ exist with $\mathcal{A}_w(T[\tau_0 + 1]) \subseteq \{M\}$. Thus we can apply Lemma 25 which allows us to deduce that $M \in \mathcal{A}_d(T[\tau])$ and in particular $\mathcal{A}_w(T[\tau]) = \emptyset$. Therefore item 1 of Definition 35 holds which allows us to conclude.

– Case 2: by definition $[\![\mathsf{out}(N', M'); P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' = [\![P', \mathcal{O}, \mathcal{I}']\!] n\mathcal{H}'\rho' \cup \{\mathcal{H}' \xrightarrow{\mathcal{O},\mathcal{I}'} \mathrm{att}_n(M'\rho')\}$. Once again, item 5 also ensures that if we denote $\mathcal{H}' = F_1 \wedge \ldots \wedge F_k \wedge \phi$ then $\sigma' \models \phi$ and there exist $\tau_1, \ldots, \tau_k$ and some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $F_1\sigma', \ldots, F_k\sigma'$ at steps $\tau_1, \ldots, \tau_k$ respectively from $\mathbb{C}(\tau_0)$ such that $\tau_j \leq \tau_0$ and $T, \mathcal{S}_p \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. Thus we can build a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step $\tau_0 + 1$ from $\mathbb{C}(\tau_0 + 1)$ as follows:

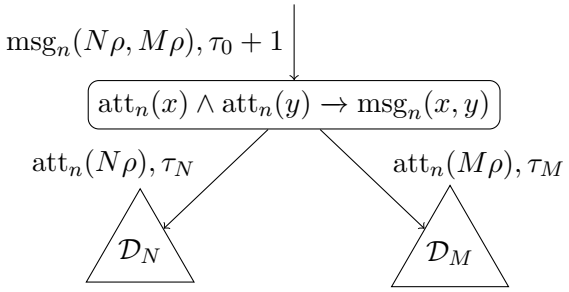


As in the previous case, $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$ meaning that $\mathcal{A}_d(T[\tau_0+1])$ and $\mathcal{A}_w(T[\tau_0+1])$ exist with $\mathcal{A}_w(T[\tau_0 + 1]) \subseteq \{M\}$. We conclude by applying Lemma 25.

- Rule I-In: In such a case, $\rho = \rho_0$, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{in}^o(N,x); Q, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(Q\{^M/_x\}, (\mathcal{O}, o), (\mathcal{I}, M\rho)\}\!\}$ with $N, M \in \mathcal{A}_0$. By our inductive hypothesis, we know that there exist $N', Q', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{in}^o(N', x); Q')\rho'\sigma' = (\mathsf{in}^o(N, x); Q)\rho, \mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{in}^o(N', x); Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. We do a case analysis on whether $N \in \mathcal{A}_I$ and $n \geq n_{IO}$.

  - Case $N \notin \mathcal{A}_I$ or $n < n_{IO}$: by definition $[\![\mathsf{in}^o(N', x); Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' = [\![Q', (\mathcal{O}, o), (\mathcal{I}', x')]\!]n(\mathcal{H}' \wedge \mathrm{msg}_n(N'\rho', x'))\rho'[x \mapsto x']$ with x' a fresh variable. Let us define $\rho'' = \rho'[x \mapsto x']$ and $\sigma'' = \sigma'[x' \mapsto M\rho]$. Since x' is fresh and x was bound (i.e. does not appear in ρ, ρ', σ'), we deduce that $Q'\rho''\sigma'' = Q'\rho'[x \mapsto x']\sigma'[x' \mapsto M\rho] = Q'\rho'\sigma'[x \mapsto M\rho] = Q\rho[x \mapsto M\rho] = Q\{^M/_x\}\rho$. Moreover, by defining $\mathcal{I}'' = (\mathcal{I}', x')$, we deduce that $\mathcal{I}''\sigma'' = (\mathcal{I}'\sigma', M\rho) = (\mathcal{I}, M\rho)$. We show that the result holds by assigning $Q', \mathcal{I}'', (\mathcal{H}' \wedge \mathrm{msg}_n(N'\rho', x')), \rho''$ and σ'' to $(Q\{^M/_x\}, (\mathcal{O}, o), (\mathcal{I}, M\rho))$. To do so, it remains to define a derivation for $\mathrm{msg}_n(N\rho, M\rho)$.

    We know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds and in particular item 1. Thus, $M, N \in \mathcal{A}_0$ implies that there exist τ_M, τ_N and some derivation $\mathcal{D}_M, \mathcal{D}_N$ of $\mathrm{att}_n(M\rho), \mathrm{att}_n(N\rho)$ at step τ_M, τ_N respectively from $\mathbb{C}(\tau_0)$ such that $\tau_M \leq \tau_0$, $\tau_N \leq \tau_0$, $T, \mathcal{S}_p \vdash \mathcal{D}_N$ and $T, \mathcal{S}_p \vdash \mathcal{D}_M$. Thus we can build a derivation $\mathcal{D}$ of $\mathrm{msg}_n(M\rho)$ at step $\tau_0 + 1$ from $\mathbb{C}(\tau_0 + 1)$ as follows:

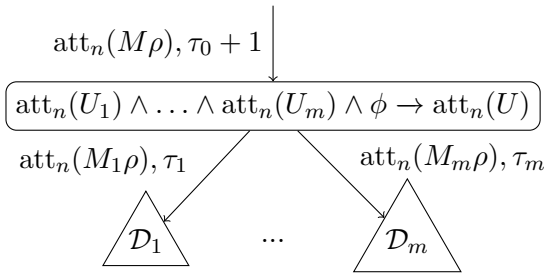
    

    Since $T, \tau_0 + 1 \vdash^{n_{IO}}_{IO} \mathrm{msg}_n(N\rho, M\rho)$, $\tau_N < \tau_0 + 1$ and $\tau_M < \tau_0 + 1$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ which allows us to conclude.

  - Case $N \in \mathcal{A}_I$ and $n \geq n_{IO}$: by definition $[\![\mathsf{in}^o(N', x); Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' = [\![Q', (\mathcal{O}, o), (\mathcal{I}', x')]\!]n(\mathcal{H}' \wedge \mathrm{att}_n(x'))\rho'[x \mapsto x']$ with x' a fresh variable. As in the previous case, we can define $\rho'' = \rho'[x \mapsto x']$, $\sigma'' = \sigma'[x' \mapsto M\rho]$ and $\mathcal{I}'' = (\mathcal{I}', x')$. Moreover, the invariant can be shown by assigning $Q', \mathcal{I}'', (\mathcal{H}' \wedge \mathrm{att}_n(x')), \rho'', \sigma''$ to $(Q\{^M/_x\}, (\mathcal{O}, o), (\mathcal{I}, M\rho))$ and defining a suitable derivation for $\mathrm{att}_n(M\rho)$. Such a derivation can be derived from the fact that $M \in \mathcal{A}_0$ and $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds (in particular item 1).

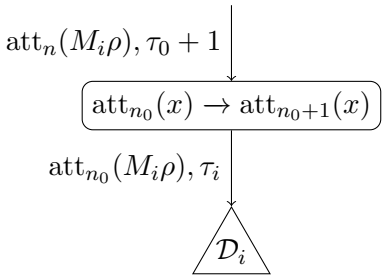
- Application of function symbol: Since $T[\tau_0] \to_c T[\tau]$ is a complete transition, we have $T[\tau_0] \xrightarrow{\text{I-App}(f, M_1, \ldots, M_m)}_i T[\tau_0 + 1] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau]$ where $f(M_1, \ldots, M_m) \Downarrow M$. In particular, $\mathcal{A}(T[\tau_0 + 1]) = \mathcal{A}_0 \cup \{M\}$ and $M_1, \ldots, M_m \in \mathcal{A}_0$. Note that $M_1, \ldots, M_m$ are terms. Thus by definition of $\Downarrow$, $f(M_1, \ldots, M_m) \Downarrow M$ implies that there exist $f(U_1, \ldots, U_m) \to U \mid\mid \phi \in \mathsf{def}(f)$ and a substitution σ such that $U_i\sigma = M_i$ for all $i \in \{1, \ldots, m\}$, $M = U\sigma$ and $\sigma \models \phi$. Moreover, since $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds and $M_1, \ldots, M_m \in \mathcal{A}_0$, we know that there exist $\tau_1, \ldots, \tau_m$ and some derivation $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of $\mathrm{att}_n(M_1\rho)$, …, $\mathrm{att}_n(M_m\rho)$ at step $\tau_1, \ldots, \tau_m$ respectively from $\mathbb{C}(\tau_0)$ such that $\tau_j \leq \tau_0$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_j$ for all

$j = 1 \dots k$. Therefore, we can build a derivation $\mathcal{D}$ of $\mathrm{att}_n(M\rho)$ at step $\tau_0 + 1$ from $\mathbb{C}(\tau_0 + 1)$ as follows:



Once again, $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}$ meaning that $\mathcal{A}_d(T[\tau_0 + 1])$ and $\mathcal{A}_w(T[\tau_0 + 1])$ exist with $\mathcal{A}_w(T[\tau_0 + 1]) \subseteq \{M\}$. We conclude by applying Lemma 25.

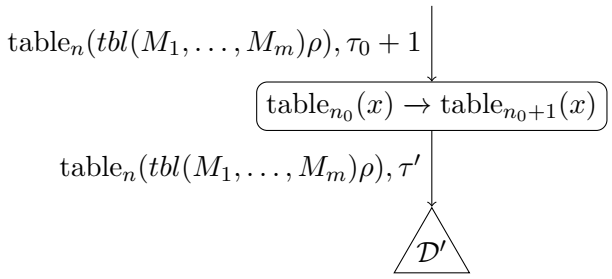
- Rule I-New: Direct by application of the clause RGen.
- Rule I-Event: In such a case, $\tau = \tau_0 + 1$, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{event}^o(ev); P, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$. By our inductive hypothesis, there exist $ev', P', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{event}^o(ev'); P')\rho'\sigma' = (\mathsf{event}^o(ev); P)\rho$, $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{event}^o(ev'); P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. By definition, $[\![\mathsf{event}^o(ev'); P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' = [\![P', \mathcal{O}, \mathcal{I}']\!]n(\mathcal{H}' \wedge \text{s-event}(o[\mathcal{I}'], ev'\rho'))\rho' \cup \{\mathcal{H}' \xrightarrow{\mathcal{O},\mathcal{I}'} \text{m-event}(o[\mathcal{I}'], ev'\rho')\}$. Note that $T, \tau \vdash^{n_{IO}}_{IO} \text{s-event}(o[\mathcal{I}], ev)$ meaning that the clause $\rightarrow \text{s-event}(o[\mathcal{I}], ev\rho) \in \mathbb{C}(\tau)$. Since $\mathcal{I} = \mathcal{I}'\sigma'$ and $ev\rho = ev'\rho'\sigma'$, we deduce that $\text{s-event}(o[\mathcal{I}], ev\rho)$ is trivially derivable from $\mathbb{C}(\tau)$ which allows us to conclude.
- Phase transition: Since $T[\tau_0] \rightarrow_c T[\tau]$ is a complete transition, $T[\tau_0] \xrightarrow{\text{I-Phase}}_i T[\tau_0 + 1] \xRightarrow[\mathsf{r}]{M_1} \ldots \xRightarrow[\mathsf{r}]{M_m} T[\tau]$ where $\mathcal{A}_0 = \{M_1, \ldots, M_m\}$. Moreover $n = n_0 + 1$ and $\mathcal{P}$ is typically $\mathcal{P}_0$ where we only remove elements. Hence, items 3, 4 and 5 of Definition 35 are trivially proved. It remains to prove items 1 and 2. By our inductive hypothesis, we know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds meaning that for all $i \in \{1, \ldots, m\}$, there exists τ_i and a derivation $\mathcal{D}_i$ of $\mathrm{att}_{n_0}(M_i\rho)$ at step $\tau_i \leq \tau_0$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$. Thus, for all $i \in \{1, \ldots, m\}$, we can build the derivation $\mathcal{D}'_i$ of $\mathrm{att}_{n_0+1}(M_i\rho)$ as follows:



Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ and $\tau_i < \tau_0 + 1$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash_w \mathcal{D}'_i$. Note that when $M_i\rho$ does not have a data constructor function symbol as root symbol, we have in fact $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_i$. Therefore, we deduce that $\mathcal{A}_w(T[\tau_0 + 1])$ and $\mathcal{A}_d(T[\tau_0 + 1])$ exist with $\mathcal{A}_w(T[\tau_0 + 1]) \subseteq \{M_1, \ldots, M_m\}$. Since $T[\tau_0 + 1] \xRightarrow[\mathsf{r}]{M_1} \ldots \xRightarrow[\mathsf{r}]{M_m} T[\tau]$, we

can apply successively Lemma 25 on $M_1, \ldots, M_m$ to obtain that $\mathcal{A}_w(T[\tau])$ and $\mathcal{A}_d(T[\tau])$ exist and $\mathcal{A}_w(T[\tau]) = \mathcal{A}_w(T[\tau_0 + 1]) \setminus \{M_1, \ldots, M_m\}$ meaning that $\mathcal{A}_w(T[\tau]) = \emptyset$ and so $\mathcal{A}_d(T[\tau]) = \mathcal{A}$. This allows us to conclude that item 1 holds.

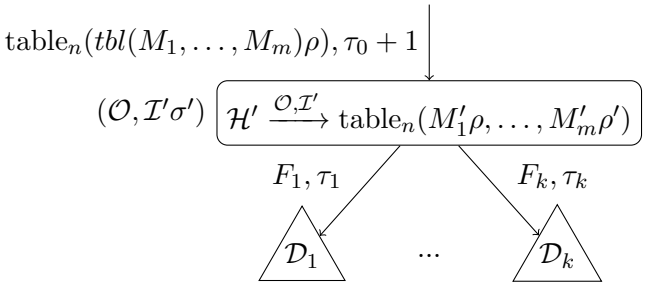
The proof of item 2 this quite similar. Indeed, by $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$, we know that for all $tbl(M_1, \ldots, M_m) \in \mathcal{T}$, there exists a derivation $\mathcal{D}'$ of $\text{table}_{n_0}(tbl(M_1, \ldots, M_m)\rho)$ at step $\tau' \leq \tau_0$. Thus, we can build a derivation $\mathcal{D}$ of $\text{table}_n(tbl(M_1, \ldots, M_m)\rho)$ with $n = n_0 + 1$ as follows:



Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and $\tau' < \tau_0 + 1$, we directly have that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ which allows us to conclude.

- Rule I-Insert: In such a case, we have $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{insert}\ tbl(M_1, \ldots, M_m); P, \mathcal{O}, \mathcal{I})\}\!\}$, $\mathcal{P} \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{T} = \mathcal{T}_0 \cup \{tbl(M_1, \ldots, M_m)\}$. By our inductive hypothesis, we know that there exists $M'_1, \ldots, M'_m, P', \mathcal{I}', \rho', \mathcal{H}', \sigma'$ such that $(\mathsf{insert}\ tbl(M'_1, \ldots, M'_m); P')\rho'\sigma' = (\mathsf{insert}\ tbl(M_1, \ldots, M_m); P)\rho$, $\mathcal{I}'\sigma' = \mathcal{I}$ and $[\![\mathsf{insert}\ tbl(M'_1, \ldots, M'_m); P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. By definition, $[\![\mathsf{insert}\ tbl(M'_1, \ldots, M'_m); P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' = [\![P', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \cup \{\mathcal{H}' \xrightarrow{\mathcal{O}, \mathcal{I}'} \text{table}_n(tbl(M_1, \ldots, M_m)\rho')\}$. By associating $P', \mathcal{I}', \rho', \mathcal{H}'$ and σ' to $(P, \mathcal{O}, \mathcal{I})$, we directly have that items 3, 4 and 5 hold. Moreover, with $\mathcal{A}_0 = \mathcal{A}$, item 1 also directly holds. Therefore, it remains to prove item 2 and in particular, we need to define a derivation for $\text{table}_n(tbl(M_1, \ldots, M_m)\rho)$.

  Since $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ holds, if we denote $\mathcal{H}' = F_1 \wedge \ldots \wedge F_k \wedge \phi$ then $\sigma' \models \phi$ and there exist $\tau_1, \ldots, \tau_k$ and some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $F_1, \ldots, F_k$ at steps $\tau_1, \ldots, \tau_k$ respectively from $\mathbb{C}(\tau_0)$ such that $\tau_j \leq \tau_0$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. Therefore, we can build a derivation $\mathcal{D}$ of $\text{table}_n(tbl(M_1, \ldots, M_m))$ at step $\tau_0 + 1 = \tau$ as follows:



Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ for all $i \in \{1, \ldots, m\}$ and $T, \tau_0 + 1 \vdash^{n_{IO}}_{IO} \text{table}_n(tbl(M_1, \ldots, M_m)\rho)$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ which allows us to conclude.

- Rule I-Get1: In such a case, we have $\mathcal{T} = \mathcal{T}_0$, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P\sigma, (\mathcal{O}, o), (\mathcal{I}, tbl(x_1, \ldots, x_m)\sigma\rho))\}\!\}$ where $tbl(x_1, \ldots, x_m)\sigma \in \mathcal{T}$ and $D\sigma \Downarrow true$. By our inductive hypothesis, we know there exist $D', P', Q', \mathcal{I}', \mathcal{H}', \rho'$ and σ' such that $D\rho = D'\rho'\sigma'$, $P\rho = P'\rho'\sigma'$, $Q\rho = Q'\rho'\sigma'$, $\mathcal{I} = \mathcal{I}'\sigma'$ and $[\![\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$.

  By definition of the destructor *equals*, we know that $D\sigma \Downarrow true$ iff $equals(D\sigma, true) \Downarrow true$ iff $equals((D\rho)(\sigma\rho), true) \Downarrow true$. Let us consider fresh variables $x'_1, \ldots, x'_m$ and the substitution $\rho_x = [x_i \mapsto x'_i]_{i=1}^m$. Hence, $D\rho(\sigma\rho) = D'\rho'\sigma'\rho_x(\rho_x^{-1}\sigma\rho)$. Since $x_1, \ldots, x_m$ were bound, we deduce that $D\rho(\sigma\rho) = D'\rho'\rho_x(\sigma'\rho_x^{-1}\sigma\rho)$. Let us denote $\alpha = \sigma'\rho_x^{-1}\sigma\rho$. By Lemma 26, $equals(D'\rho'\rho_x, true)\alpha \Downarrow true$ implies that there exist U', σ_1, σ'_1 and ϕ such that $equals(D'\rho'\rho_x, true) \Downarrow' (U', \sigma_1, \phi)$, $U = U'\sigma'_1$, $\alpha = (\sigma_1\sigma'_1)_{|dom(\alpha)}$ and $\sigma'_1 \models \phi$. Note that by definition of *equals*, we have either $U' = \mathsf{fail}$ or $U' = true$. Since $U = U'\sigma'_1$, we deduce that $U' = true$.

  Let us denote $\rho'' = \rho'\rho_x\sigma_1$, $\sigma'' = \sigma'_1$ and $\mathcal{I}'' = (\mathcal{I}'\sigma_1, tbl(x_1, \ldots, x_m)\rho_x\sigma_1)$. We show that $\mathcal{I}''\sigma'' = (\mathcal{I}, tbl(x_1, \ldots, x_m)\sigma\rho)$ and $P\sigma\rho = P'\rho''\sigma''$: We know that $\alpha = (\sigma_1\sigma'_1)_{|dom(\alpha)}$ hence $\mathcal{I}'\sigma_1\sigma'_1 = \mathcal{I}'\alpha = \mathcal{I}'\sigma'\rho_x^{-1}\sigma\rho = \mathcal{I}\rho_x^{-1}\sigma\rho$. Note that $\mathcal{I}$ is closed and does not contain any name (contains only patterns). Moreover, $dom(\rho_x^{-1}\sigma) = \{x_1, x'_1, \ldots, x_m, x'_m\}$. Thus, $\mathcal{I}\rho_x^{-1}\sigma\rho = \mathcal{I}$. Similarly, we have $tbl(x_1, \ldots, x_m)\rho_x\sigma_1\sigma'' = tbl(x_1, \ldots, x_m)\rho_x\alpha = tbl(x_1, \ldots, x_m)\rho_x\sigma'\rho_x^{-1}\sigma\rho$. Note that $dom(\sigma') \cap \{x_1, x'_1, \ldots, x_m, x'_m\} = \emptyset$ since $x_1, \ldots, x_m$ were bound and $x'_1, \ldots, x'_m$ are fresh. Thus, $tbl(x_1, \ldots, x_m)\rho_x\sigma_1\sigma'' = tbl(x_1, \ldots, x_m)\sigma'\sigma\rho$ and so $tbl(x_1, \ldots, x_m)\rho_x\sigma_1\sigma'' = tbl(x_1, \ldots, x_m)\sigma\rho$. This allows us to conclude that $\mathcal{I}''\sigma'' = (\mathcal{I}, tbl(x_1, \ldots, x_m)\sigma\rho)$.

  Let us now prove $P\sigma\rho = P'\rho''\sigma''$: $P'\rho''\sigma'' = P'\rho'\rho_x\sigma_1\sigma'_1$. Since $(\sigma_1\sigma'_1)_{|dom(\alpha)} = \alpha$, we deduce that $P'\rho''\sigma'' = P'\rho'\rho_x\sigma'\rho_x^{-1}\sigma\rho$. Once again, we can swap the substitution to obtain $P'\rho''\sigma'' = P'\rho'\sigma'\sigma\rho = P\rho\sigma\rho = P\sigma\rho$.

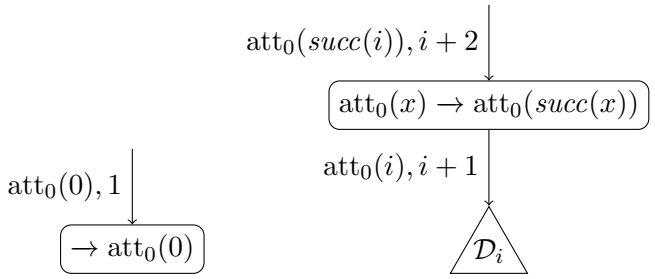
  By definition of $[\![\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho'$, since $equals(D'\rho'\rho_x, true) \Downarrow' (U', \sigma_1, \phi)$, we have $[\![P, (\mathcal{O}, o), (\mathcal{I}'\sigma_1, tbl(x_1, \ldots, x_m)\rho_x\sigma_1)]\!]n(\mathcal{H}'\sigma_1 \wedge \mathrm{table}_n(tbl(x_1, \ldots, x_m)\rho_x\sigma_1)) \wedge \phi)(\rho'\rho_x\sigma_1) \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. Therefore, by associating $P', \mathcal{I}'', \rho'', (\mathcal{H}'\sigma_1 \wedge \mathrm{table}_n(tbl(x_1, \ldots, x_m)\rho_x\sigma_1)) \wedge \phi)$ and σ'' to $(P\sigma, (\mathcal{O}, o), (\mathcal{I}, tbl(x_1, \ldots, x_m)\sigma\rho))$, we already prove items 3 and 4 of Definition 35 for $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$. Since $\mathcal{A} = \mathcal{A}_0$ and $\mathcal{T} = \mathcal{T}_0$, it only remains to prove item 5.

  First, we know that $\sigma'_1 \models \phi$ and so $\sigma'' \models \phi$. Second, notice that $\mathcal{H}'\sigma_1\sigma'_1 = \mathcal{H}'\alpha = \mathcal{H}\sigma'\rho_x^{-1}\sigma\rho$. Considering that $dom(\rho_x^{-1})$ and $dom(\sigma)$ are respectively variables $x'_1, \ldots, x'_m$ and $x_1, \ldots, x_m$, we deduce that $\mathcal{H}'\sigma_1\sigma'_1 = \mathcal{H}'\sigma'\rho$. Since $\mathcal{H}'$ and σ' do not contain names (only patterns), we conclude that $\mathcal{H}'\sigma_1\sigma'_1 = \mathcal{H}'\sigma'$. Thus, to prove item 5, we only need to define a derivation for $\mathrm{table}_n(tbl(x_1, \ldots, x_m)\rho_x\sigma_1\sigma'')$. However, we already proved that $tbl(x_1, \ldots, x_m)\rho_x\sigma_1\sigma'' = tbl(x_1, \ldots, x_m)\sigma\rho$ and we know by hypothesis that $tbl(x_1, \ldots, x_m)\sigma \in \mathcal{T}$. Thus, since $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_0)$ and in particular item 2, we can conclude.

- Rule I-Get2: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathcal{O}, \mathcal{I})\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(Q, \mathcal{O}, \mathcal{I})\}\!\}$. By our inductive hypothesis, we know that there exist $D', P', Q', \mathcal{I}', \mathcal{H}', \rho'$ and σ' such that $D\rho = D'\rho'\sigma'$, $P\rho = P'\rho'\sigma'$, $Q\rho = Q'\rho'\sigma'$, $\mathcal{I} = \mathcal{I}'\sigma'$ and $[\![\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$. But by definition $[\![Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho' \subseteq [\![\mathsf{get}^o\ tbl(x_1, \ldots, x_m)\ \mathsf{suchthat}\ D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q', \mathcal{O}, \mathcal{I}']\!]n\mathcal{H}'\rho'$. Hence we can directly conclude by associating $Q', \mathcal{H}', \rho'$ and σ' to $(Q, \mathcal{O}, \mathcal{I})$.

- Initial transition: The first complete transition build the natural numbers in the attacker

knowledge, i.e. $T[0] \xrightarrow{\text{I-App}(0)}_i \xrightarrow{\text{I-App}(succ,0)}_i \ldots \xrightarrow{\text{I-App}(succ,n-1)}_i T[\tau]$ with $\tau = n+1$. Thus, $\mathcal{A} = \{0, 1, \ldots, n\}$. We can build the derivations $\mathcal{D}_0, \ldots, D_n$ of $\text{att}_0(0), \ldots, \text{att}_0(n)$ at step $1, \ldots, n+1$ respectively as follows:



On the left is the derivation $\mathcal{D}_0$ and on the right is the derivation $\mathcal{D}_{i+1}$ for all $i \in \{0, \ldots, n-1\}$. Notice for all $i \in \{0, \ldots, n\}$, $T, i+1 \vdash^{n_{IO}}_{IO} \text{att}_0(i)$ and so we obtain that for all $i \in \{0, \ldots, n-1\}$, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ which allows us to conclude. □

B.3 Main proof

Theorem 1. *Let $\mathcal{C}_I = \rho_0, P_0, \mathcal{A}_0$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\kappa_{io} \in \mathbb{N}$.*

For all $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \rightarrow_i)$, for all ground facts F different from a sure-event, for all steps τ, if $T, \tau \vdash^{\kappa_{io}}_{IO} F$ then there exists a derivation $\mathcal{D}$ of F at step τ from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, \kappa_{io}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$.

Proof. Consider a trace $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. Let F be a ground fact different from a sure-event and a step τ such that $T, \tau \vdash^{n_{IO}}_{IO} F$. For all τ, let us denote $\mathbb{C}(\tau) = \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}^{<\tau}_e(T)$. Let us denote $T[\tau] = n, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$.

We do a case analysis on the fact F:

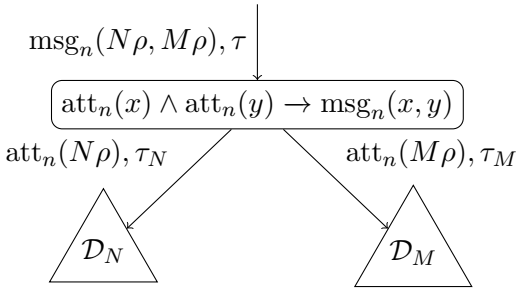
- Case $F = \text{att}_n(M')$: By definition of $T, \tau \vdash^{n_{IO}}_{IO} F$, we deduce that $M' = M\rho$ for some $M \in \mathcal{A}$. We first need to consider whether τ is a complete step or not. If τ is a complete step then by Lemma 27, we know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$. Thus by item 1 of Definition 35, we know that there exists $\tau' \leq \tau$ and a derivation $\mathcal{D}$ of F at step τ' from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}^{\leq \tau'}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. Since F is not being an event fact and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$, then for all edges labeled with an event fact and a step τ'', we have $\tau'' < \tau'$ (otherwise item 3 of Definition 18 would not satisfied). Thus, we conclude that $\mathcal{D}$ is also a derivation from $\mathbb{C}(\tau') \subseteq \mathbb{C}(\tau)$. Finally, by replacing the label of the incoming edge of the root F, τ' by F, τ, we obtain a new derivation $\mathcal{D}'$ of F at step τ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$.

  If τ is not a complete step then there exist τ_1, τ_2 such that $T[\tau_1] \rightarrow_c T[\tau_2]$, τ_1, τ_2 are both complete steps and $\tau_1 < \tau < \tau_2$. But by Lemma 27, we know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_2)$ holds. Hence there exist $\tau' \leq \tau_2$ and a derivation $\mathcal{D}$ of F at step τ' from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}^{\leq \tau'}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. Moreover, we also know that τ' is the smallest step such that $T, \tau' \vdash^{n_{IO}}_{IO} F$. Hence $\tau' \leq \tau$. Once again, we conclude by replacing the label of the incoming edge of the root F, τ' by F, τ.

- Case $F = \text{table}_n(tbl(M'_1, \ldots, M'_m))$: By definition of $T, \tau \vdash_{IO}^{n_{IO}} F$, we deduce that there exist $M_1, \ldots, M_m$ such that $tbl(M_1, \ldots, M_m) \in \mathcal{T}$ and $tbl(M_1, \ldots, M_m)\rho = tbl(M'_1, \ldots, M'_m)$. As for the previous case, either τ is a complete step or is between two complete steps. Since $\tau \neq 0$ (the initial set of inserted elements in $\mathcal{C}_I$ is empty), we deduce that there exist two complete steps τ_1, τ_2 such that $T[\tau_1] \rightarrow_c T[\tau_2]$ and $\tau_1 < \tau \leq \tau_2$. By Lemma 27, we know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau_2)$ holds. Hence there exist $\tau' \leq \tau_2$ and a derivation $\mathcal{D}$ of F at step τ' from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}_{\vec{e}}^{\leq \tau'}(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. Moreover, we also know that τ' is the smallest step such that $T, \tau' \vdash_{IO}^{n_{IO}} F$. Hence $\tau' \leq \tau$. Since F is not an event fact, we can conclude by replacing the label of the incoming edge of the root F, τ' by F, τ.

- Case $F = \text{msg}_n(N', M')$: By definition of $T, \tau \vdash_{IO}^{n_{IO}} F$, we deduce that there exist N, M such that $N' = N\rho$, $M' = M\rho$, $T[\tau - 1] \xrightarrow{\text{msg}(N\rho, M\rho)}_i T[\tau]$ and if the rule applied is I-Out then $n < n_{IO}$ or $N \notin \mathcal{A}_I$. Let us do a case analysis on the rule applied:

  - Case I-Msg: In such a case, $M, N \in \mathcal{A}(T[\tau - 1]) \subseteq \mathcal{A}(T[\tau])$. Moreover, τ is a complete step (note that $\tau - 1$ is not necessarily a complete step). Thus by Lemma 27, we know that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$ and so there exist two derivations $\mathcal{D}_M, \mathcal{D}_N$ of $\text{att}_n(M\rho), \text{att}_n(N\rho)$ at steps $\tau_M, \tau_N \leq \tau$ from $\mathbb{C}(\tau)$ respectively such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_M$, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_N$ and τ_M (resp. τ_N) is the smallest step such that $T, \tau_M, n_{IO} \vdash \text{att}_n(M\rho)$ (resp. $T, \tau_N, n_{IO} \vdash \text{att}_n(N\rho)$). Since the τ-th rule is not I-App, we deduce that $\tau_N < \tau$ and $\tau_M < \tau$. Therefore, we conclude by building the following derivation $\mathcal{D}$ of F at step τ:

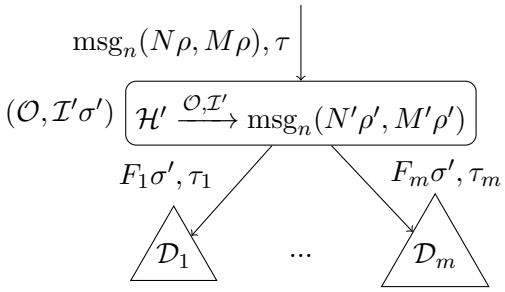


  - Case I-I/O: In such a case, $\tau - 1$ and τ are complete steps and so $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$ and $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau - 1)$ hold. Moreover, since $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$, we deduce that $n < n_{IO}$ or $N \notin \mathcal{A}_I$. Finally, by definition of the rule I-I/O, $T[\tau - 1] = n, \rho, \mathcal{P}', \mathcal{T}, \mathcal{A}, \Lambda$ where $\mathcal{P}' = \mathcal{P}'' \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}_1, \mathcal{I}_1), (\mathsf{in}^o(N, x); Q, \mathcal{O}_2, \mathcal{I}_2)\}\!\}$ and $\mathcal{P} = \mathcal{P}'' \cup \{\!\{(P, \mathcal{O}_1, \mathcal{I}_1), (Q\{^M/_x\}, (\mathcal{O}_2, o), (\mathcal{I}_2, M\rho))\}\!\}$. By $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau - 1)$ and $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau)$ (specifically items 3 and 4), we know there exists $Q', \mathcal{I}'_2, \sigma', \rho', N''$ and $\mathcal{H}'$ such that $N''\rho'\sigma' = N\rho$, $[\![Q', \mathcal{O}, (\mathcal{I}'_2, x')]\!] n(\mathcal{H}' \wedge \text{msg}_n(N''\rho', x'))\rho' \subseteq \mathbb{C}(\tau)$ and $(\mathcal{I}_2, M\rho) = (\mathcal{I}'_2\sigma', x'\sigma')$. Furthermore by item 5, we know that there exists $\tau' \leq \tau$ and a derivation $\mathcal{D}$ of $\text{msg}_n(N''\rho'\sigma', x'\sigma')$ at step τ' from $\mathbb{C}(\tau)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. Since $\text{msg}_n(N''\rho'\sigma', x'\sigma') = \text{msg}_n(N\rho, M\rho)$, we can conclude by replacing the label of the incoming edge of the root F, τ' by F, τ.

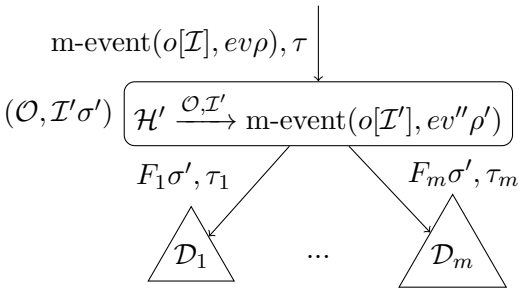
  - Case I-In: Same proof as for the rule I-Msg since $M, N \in \mathcal{A}(T[\tau - 1])$.

  - Case I-Out: In such a case, $n < n_{IO}$ or $N \notin \mathcal{A}_I$. Moreover, $\tau - 1$ is a complete step and $T[\tau - 1] = n, \rho, \mathcal{P}', \mathcal{T}, \mathcal{A}', \Lambda$ where $\mathcal{P}' = \mathcal{P}'' \cup \{\!\{(\mathsf{out}(N, M); P, \mathcal{O}, \mathcal{I})\}\!\}$ and

$\mathcal{P} = \mathcal{P}'' \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$. $\tau-1$ being a complete step implies that $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau-1)$ holds. Thus by items 3 and 4, we deduce that there exist $N'', M'', \mathcal{H}'\sigma', \rho'$ and $\mathcal{I}'$ such that $\mathcal{H}' \xrightarrow{\mathcal{O},\mathcal{I}'} \text{msg}_n(M''\rho', N''\rho') \in \mathbb{C}(\tau-1)$ and $M''\rho'\sigma' = M\rho$ and $N''\rho'\sigma' = N\rho$. By item 5, we also know that if we denote $\mathcal{H}' = F_1 \wedge \ldots \wedge F_m \wedge \phi$ then $\sigma' \models \phi$ and there exists $\mathcal{D}_1, \ldots, \mathcal{D}_m$ derivations of $F_1\sigma', \ldots, F_m\sigma'$ at steps $\tau_1, \ldots, \tau_m$ respectively from $\mathbb{C}(\tau)$ such $\tau_i \leq \tau - 1$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ for all $i \in \{1, \ldots, m\}$. Therefore, we can build the following derivation of $\text{msg}_n(N\rho, M\rho)$ at step τ to conclude.



- Case $F = \text{m-event}(o', ev')$: By definition of $T, \tau \vdash^{n_{IO}}_{IO} F$, we deduce $T[\tau-1] \xrightarrow{\text{event}(o',ev')}_i T[\tau]$ where $T[\tau-1] = n, \rho, \mathcal{P}', \mathcal{T}, \mathcal{A}, \Lambda$, $\mathcal{P}' = \mathcal{P}'' \cup \{\!\{(\mathsf{event}^o(ev); P, \mathcal{O}, \mathcal{I})\}\!\}$, $\mathcal{P} = \mathcal{P}'' \cup \{\!\{(P, \mathcal{O}, \mathcal{I})\}\!\}$, $o' = o[\mathcal{I}]$ and $ev' = ev\rho$. Moreover, we know that $\tau - 1$ is a complete step and so $\mathcal{I}nv_{\mathcal{S}_p}(T, \tau-1)$ holds. By items 3 and 4 of Definition 35, we deduce that there exists $ev'', \mathcal{H}', \sigma', \rho'$ and $\mathcal{I}'$ such that $\mathcal{H}' \xrightarrow{\mathcal{O},\mathcal{I}'} \text{m-event}(o[\mathcal{I}'], ev''\rho') \in \mathbb{C}(\tau-1)$ and $\mathcal{I}'\sigma' = \mathcal{I}$ and $ev''\rho'\sigma' = ev\rho$. By item 5, we also know that if we denote $\mathcal{H}' = F_1 \wedge \ldots \wedge F_m \wedge \phi$ then $\sigma' \models \phi$ and there exists $\mathcal{D}_1, \ldots, \mathcal{D}_m$ derivations of $F_1\sigma', \ldots, F_m\sigma'$ at steps $\tau_1, \ldots, \tau_m$ respectively from $\mathbb{C}(\tau)$ such $\tau_i \leq \tau - 1$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ for all $i \in \{1, \ldots, m\}$. Therefore, we can build the following derivation of $\text{m-event}(o[\mathcal{I}], ev\rho)$ at step τ to conclude.



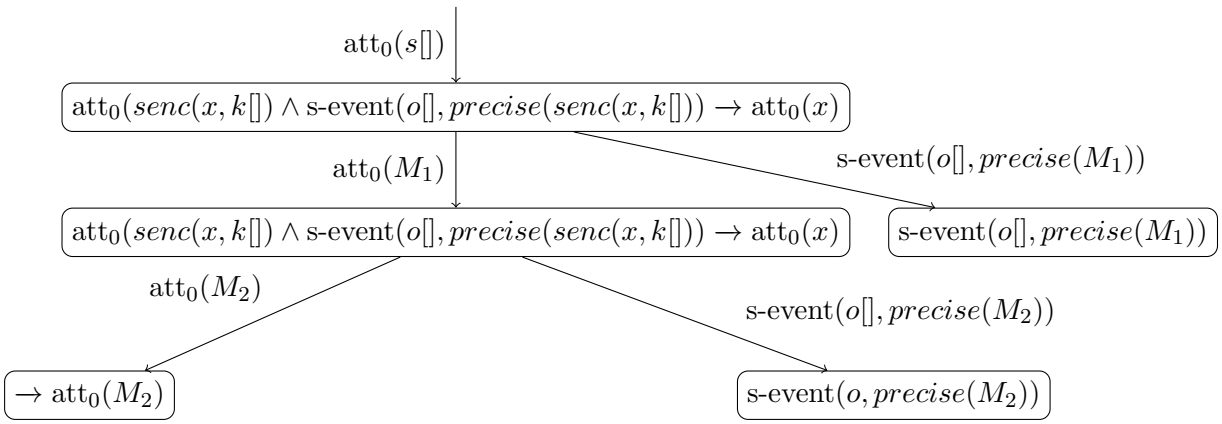
□

C Proof of Theorem 2

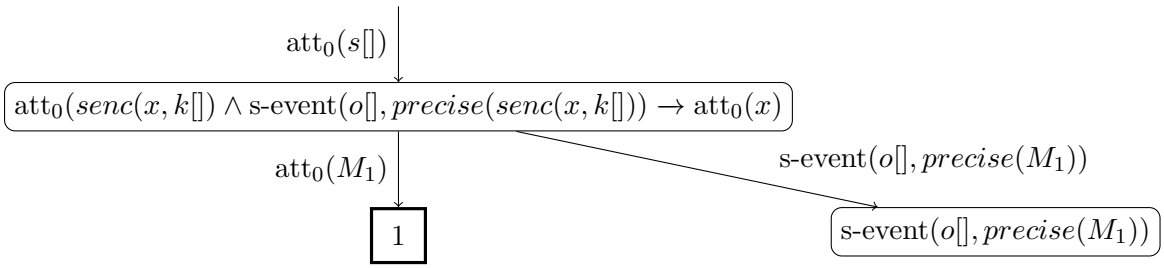
In this section, we show the soundness of all the transformations applied on the set of clauses during the saturation procedure. To simplify the reading of the proof, we introduce the notion of *derivation context* which is typically a derivation where leaves can be *holes*. Similarly to a term context, we will use the notation $\mathcal{D}[\__1, \ldots, \__n]$ to denote a derivation context $\mathcal{D}$ with

n holes. For some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_n$, we denote $\mathcal{D}[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ the derivation obtained by replacing each hole $\__i$ in $\mathcal{D}$ by $\mathcal{D}_i$. Note that for $\mathcal{D}[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ to be a derivation, it is required that for all $i \in \{1, \ldots, n\}$, the incoming edge of the root of $\mathcal{D}_i$ has the same label as the incoming edge of the hole $\__i$ in $\mathcal{D}$.

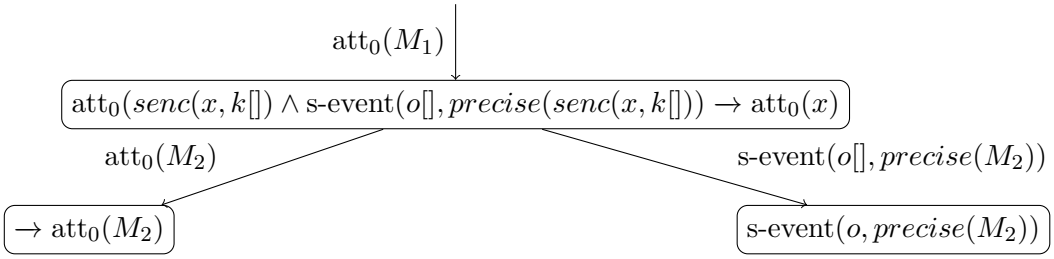
Example 28. In Example 18, we provided the following small derivation, denoted $\mathcal{D}$, describing the effects of precise events:



Below we show the derivation context $\mathcal{D}'[\__1]$ obtained by replacing the subtree rooted by $\text{att}_0(\mathit{senc}(x, k[]) \rightarrow \text{att}_0(x)$ with a hole. Graphically, the hole is represented by a numbered square.

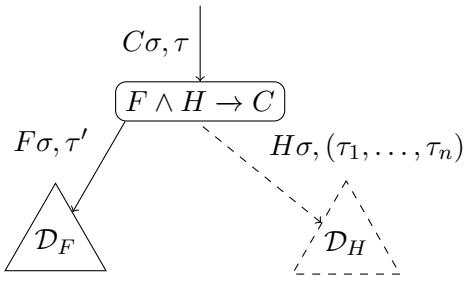


When the context contains only one hole, we may remove the number 1 from the square and directly denote the context $\mathcal{D}'[\_]$. If we now consider the following derivation $\mathcal{D}_1$, we obtain $\mathcal{D} = \mathcal{D}'[\mathcal{D}_1]$:



▶

In previous sections, we used triangle nodes to graphically represent derivations. In this section, we will also use dashed triangle nodes to graphically represent conjunction of derivations. For example, the following is a graphical representation of a derivation of a fact $C\sigma$ at step τ. The root is labeled by the clause $F \wedge H \to C$. $\mathcal{D}_F$ represents the derivation of $F\sigma$ at step τ'. H is a conjunction of n facts and possibly a constraint formula. $\mathcal{D}_H$ here represents the n derivations of the n facts of $H\sigma$ respectively at step $\tau_1, \ldots, \tau_n$. We may omit the labels on the edges when they are not relevant for the proof, for the sake of readability.



C.1 Preamble

To show the soundness of the derivation, we also define the *size* of a derivation, the notion of *basic derivation of natural numbers* as well as the notion of *(strictly) selection free derivation.*

Definition 36. *Given $k \in \mathbb{N}$, we say that a derivation $\mathcal{D}$ is a basic derivation of k if there exists $n \in \mathbb{N}$ such that $\mathcal{D}$ is the derivation with a unique leaf labeled by $\to \mathrm{att}_n(0)$ and all internal nodes (including the root) are labeled by the clause $\mathrm{att}_n(x) \to \mathrm{att}_n(succ(x))$.*

In the rest of this section, we will call *clauses for natural numbers* the clauses $\to \mathrm{att}_n(0)$ and $\mathrm{att}_n(x) \to \mathrm{att}_n(succ(x))$.

Definition 37. *Let $\mathcal{D}$ be a derivation. We define the size of $\mathcal{D}$, denoted $|\mathcal{D}|$, as the number of nodes η in $\mathcal{D}$ such that:*

- *η is not labeled by a clause $\to \text{s-event}(o, ev)$ for all o, ev; and*
- *the sub-derivation rooted by η is not a basic derivation of some natural number.*

We say that $\mathcal{D}$ is selection free *(resp.* strictly selection free*) when for all all nodes (resp. all nodes different from the root) of $\mathcal{D}$ labeled by $H \to C$, $\mathrm{sel}(H \to C) = \emptyset$.*

We also show that the set of clauses we initially generate from the protocols and the subsequent sets of clauses obtained through the saturation procedure are well originated.

Lemma 28. *Let $\mathcal{C}_I = \rho_0, P_0, \mathcal{A}_0$ be an initial instrumented configuration. Let $n \in \mathbb{N}$. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}, \mathcal{L}_i$ be two sets of* ProVerif *lemmas.*

The sets $\mathbb{C} = \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n) \cup \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$ and $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ are well originated.

Proof. For $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$, the proof directly holds by a quick look at the clauses. Notice that for the clause (Rf), the rewrite rules generated by ProVerif ensure that all variables in the right-hand side of the rewrite rule occur in the left-hand side of the rule.

For $\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n)$, the clauses being generated from a closed process, variables are only bound and introduced by either an input or a table lookup. In both cases, the bound variables occur

in the hypotheses within a fact that satisfies the origination property. The term evaluation $x = D$ also binds the variable x but that is directly replaced by the result of the evaluation of D when generating the clauses. The second item of the well originated property is satisfied by definition of the translation of $[\![\mathsf{phase}\ n'; P, \mathcal{O}, \mathcal{I}]\!] n\mathcal{H}\rho$ since clauses are only generated when $n' > n$.

To prove that $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ is well originated, it suffices to show that the application of any of the transformation rules on a well originated set of clauses results in a well originated set of clauses. Most of the rules are either applying a substitution on a clause preserving the origination property or removing a clause completely from the set. There are two particular cases: the resolution rule and the application of lemmas. In the former case, (using the notation of Section 5), notice that if F was a fact $\mathrm{att}_n(M)$, $\mathrm{msg}_n(N, M)$ or $\mathrm{table}_n(M)$ then all the variables in $F\sigma$ would occur accordingly in $H\sigma$ since $F\sigma = C\sigma$ and $H \to C$ is well originated. In the latter case, the lemmas can indeed introduce new variables that do not occur anywhere else (typically the existential variables in the lemma). However a lemma can only add events and formulas in the hypothesis of the clause which are not considered in the origination property. □

Property on ProVerif lemmas For the proof of both Theorem 2 and for the proof of Theorem 4, we will need to show the soundness of the transformation rules that apply lemmas and inductive lemmas. We show below a generic result that will be reused in both proofs.

Lemma 29. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of* ProVerif *IO-n_{IO}-compliant lemmas. Let $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$ and $\mathcal{M}$ be a multiset such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \mathcal{M})$ holds. Let $\psi \in \mathcal{L} \cup \mathcal{L}_i$ such that $\psi = (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} E_{1,j} \wedge \ldots \wedge E_{\ell_j,j} \wedge \phi_j)$.*

For all substitutions σ, for all steps $\tau_1, \ldots, \tau_n$, if $T, \tau_1 \vdash^{n_{IO}}_{IO} F_1\sigma$, $\ldots$, $T, \tau_n \vdash^{n_{IO}}_{IO} F_n\sigma$ and either $\psi \in \mathcal{L}$ or $\{\!\{\tau_1, \ldots, \tau_n\}\!\} <_m \mathcal{M}$ then there exist $j \in \{1, \ldots, m\}$, a substitution σ' and $\tau'_1, \ldots, \tau'_{\ell_j}$ steps such that:

- *for all $i \in \{1, \ldots, n\}$, $F_i\sigma = F_i\sigma'$*
- *for all $k \in \{1, \ldots, \ell_j\}$, $\tau'_k \leq \max(\tau_1, \ldots, \tau_n)$ and $T, \tau'_k \vdash^{n_{IO}}_{IO} E_{k,j}\sigma'$*
- *$\vdash^{n_{IO}}_{IO} \phi_j\sigma'$*

Proof. By hypothesis, we know that $T, \tau_1 \vdash^{n_{IO}}_{IO} F_1\sigma$, $\ldots$, $T, \tau_n \vdash^{n_{IO}}_{IO} F_n\sigma$. Hence, let us consider the trace T'' prefix of T such that $\mathsf{max}_{step}(T'') = \max(\tau_1, \ldots, \tau_n)$. To be able to apply the properties given by $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \mathcal{M})$, we would need to ensure that $T'' \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$ but that is not true in general for all prefixes of T. We show however that we can find a prefix $T' \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$ of T such that T'' is also a prefix of T' and that all transitions applied between T'' and T' are applications of either the rule I-Msg or I-App.

Let us denote $\tau'' = \mathsf{max}_{step}(T'')$. Let us assume that $T'' \notin \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$. Since $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$, we deduce from Definitions 12 and 13 that there exist $\tau_0 \leq \tau'' < \tau'_0$ such that $T[\tau_0] = n_0, \rho, \mathcal{P}, \mathcal{T}, \mathcal{A}, \Lambda$ and one of the following properties holds

- $T[\tau_0 - 1] \to_i T[\tau_0]$ by the rule I-Phase, $\mathcal{A} = \{M_i\}_{i=1}^{m_0}$ and $T[\tau_0] \xrightarrow[\mathsf{r}]{M_1} \ldots \xrightarrow[\mathsf{r}]{M_{m_0}} T[\tau'_0]$.
- there exists $M \in \mathcal{A} \setminus \mathsf{data}(T, \tau_0)$ and $T[\tau_0] \xRightarrow[\mathsf{dr}]{M} T[\tau'_0]$.

- $\tau'_0 = k+1$ where k is the greatest integer occurring in T and $T[0] \xrightarrow{\text{I-App}(0)}_i \overset{1}{\underset{\mathsf{r}}{\Rightarrow}} \ldots \overset{k}{\underset{\mathsf{r}}{\Rightarrow}} T[k+1]$
- $T[\tau_0 - 1] \xrightarrow{\text{msg}(N,M)}_i T[\tau_0]$ by the rule I-Out, $j \geq n$, $N \in \mathcal{A}_0$ and $T[\tau_0] \overset{M}{\underset{\mathsf{dr}}{\Longrightarrow}} T[\tau'_0 - 1] \xrightarrow{\text{msg}(N,M)}_i T[\tau'_0]$ by the rule I-Msg.

Notice that between the steps τ'' and τ'_0, only rules I-Msg or I-App are applied. Thus, amongst the possible pairs (τ_0, τ'_0) satisfying the above conditions, if we take the one with the biggest τ'_0 and the prefix T' of T with $\mathsf{max}_{step}(T') = \tau'_0$, we therefore obtain that $T' \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. Finally, since between steps τ'' and τ'_0, only rules I-Msg or I-App are applied, we deduce that for all events E, for all steps τ_E, if $T', \tau_E \vdash^{n_{IO}}_{IO} E$ then $\tau_E \leq \tau''$ $(\star)$.

Let us now prove the main property of the lemma. Let us do a case analysis on whether $\psi \in \mathcal{L}$ or not:

- If $\psi \in \mathcal{L}$ then we know from $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \mathcal{M})$ that $(\vdash^{n_{IO}}_{IO}, \vdash_i, \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)) \models \psi$. In particular, we have $(\vdash^{n_{IO}}_{IO}, \vdash_i, \{T'\}) \models \psi$. Hence there exist an annotated formula Φ w.r.t. (n, τ'_0) and a substitution σ' such that $\overline{\Phi} = \bigvee_{j=1}^{m} E_{1,j} \wedge \ldots \wedge E_{\ell_j,j} \wedge \phi_j$; $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T', (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$.
- If $\psi \notin \mathcal{L}$ then we know that $\psi \in \mathcal{L}_i$ and $\{\!\{\tau_1, \ldots, \tau_n\}\!\} <_m \mathcal{M}$ by our hypotheses. Since $|T'| \leq |T|$ (T' is a prefix of T), we deduce that $(T', \{\!\{\tau_1, \ldots, \tau_n\}\!\}) <_{ind} (T, \mathcal{M})$. Therefore, we obtain that $\mathcal{IH}_\psi(T', \{\!\{\tau_1, \ldots, \tau_n\}\!\})$ holds. By definition, it allows us to deduce that once again, there exist an annotated formula Φ w.r.t. (n, τ'_0) and a substitution σ' such that $\overline{\Phi} = \bigvee_{j=1}^{m} E_{1,j} \wedge \ldots \wedge E_{\ell_j,j} \wedge \phi_j$; $F_i\sigma = F_i\sigma'$ for $i = 1 \ldots n$ and $(\vdash_i, T', (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$.

In both cases, $(\vdash_i, T', (\tau_1, \ldots, \tau_n, \sigma)) \models \Phi\sigma'$ implies the existence of $j \in \{1, \ldots, m\}$ and some partial functions $\mu_1, \ldots, \mu_{\ell_j}$ such that for all $k \in \{1, \ldots, \ell_j\}$, $T', \mu_k(\tau_1, \ldots, \tau_n, \sigma) \vdash_i E_{k,j}\sigma'$ and $\vdash_i \phi_j\sigma'$. Note that since $E_{1,j}, \ldots, E_{\ell_j,j}$ are events and ϕ_j is a formula, we deduce that $T', \mu_k(\tau_1, \ldots, \tau_n, \sigma) \vdash^{n_{IO}}_{IO} E_{k,j}\sigma'$ and $\vdash^{n_{IO}}_{IO} \phi_j\sigma'$. Moreover, thanks to property $(\star)$ that we previously showed, we also deduce that $\mu_k(\tau_1, \ldots, \tau_n, \sigma) \leq \tau'' = \max(\tau_1, \ldots, \tau_n)$. Denoting $\mu_k(\tau_1, \ldots, \tau_n, \sigma)$ by τ'_k allows us to conclude. □

Clauses from $\mathbb{C}_{std}$ preserved by saturation. To prove the soundness of the saturation procedure, we will rely on the fact that the clauses from $\mathbb{C}_{std}$ are always contained in our main set of clauses. Hence, we need to show that the saturation procedure does not alter nor remove them.

Lemma 30. *Let $\mathbb{C}$ be a set of well-originated selection-free clauses. Assume that $\mathbb{C} = \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})(\mathbb{C})$. Let $\mathcal{D}$ be a partial derivation of some fact F from $\mathbb{C}$. For all derivation contexts $\mathcal{D}_c$ and partial derivations $\mathcal{D}'$ such that $\mathcal{D} = \mathcal{D}_c[\mathcal{D}']$, if the root's incoming edge of $\mathcal{D}'$ is labeled by $\text{att}_n(x)$ for some variable x, then $\mathcal{D}'$ contains a branch that is a sequential application of clauses $\text{att}_i(y) \rightarrow \text{att}_{i+1}(y)$ and there exists $j \leq n$ such that $\text{att}_j(y) \in \mathbb{F}_{us}(\mathcal{D})$.*

Proof. We prove this result by induction on the size of $\mathcal{D}'$. In the base case, $\mathcal{D}'$ is in fact a leaf. If the leaf is unlabeled, then the result directly holds since incoming edges of unlabeled leaves are in $\mathbb{F}_{us}(\mathcal{D})$. If the leaf is labeled, then it must be labeled by a clause R of the form $\rightarrow C$ such that $R \sqsupseteq \rightarrow \text{att}_n(x)$. Thus there exists σ such that $C\sigma = \text{att}_n(x)$ and so $C = \text{att}_n(y)$ for

some variable y. However, R must be a clause from $\mathbb{C}$ meaning that R must be well originated which is in contradiction with the fact that $C = \mathrm{att}_n(y)$ and R does not have hypotheses.

In the inductive step, we know that the child of the root of $\mathcal{D}'$ is labeled by a clause $R = (H \to C) \in \mathbb{C}$. Furthermore, if $H_1, \ldots, H_m$ are the labels of the outgoing edges of the child of the root of $\mathcal{D}'$, then $R \sqsupseteq H_1 \wedge \ldots \wedge H_m \to \mathrm{att}_n(x)$. Hence there exists σ such that $C\sigma = \mathrm{att}_n(x)$ and $H\sigma \subseteq_m \{\!\{H_1, \ldots, H_m\}\!\}$. $C\sigma = \mathrm{att}_n(x)$ implies that $C = \mathrm{att}_n(y)$ for some variable y. Note that R is by hypothesis selection free since $R \in \mathbb{C}$. Hence, with $\mathrm{att}_n(y)$ being unselectable, we deduce from Definition 19 that all facts in H are unselectable. Furthermore, R is also well-originated. Hence by Definition 21, we deduce that there exists $n' \leq n$ such that $\mathrm{att}_{n'}(y) \in H$. By definition of a partial derivation (last item), we obtain that R is in fact the clause $\mathrm{att}_{n-1}(y) \to \mathrm{att}_n(y)$. Since $R \sqsupseteq H_1 \wedge \ldots \wedge H_m \to \mathrm{att}_n(x)$, we deduce that $H_j = \mathrm{att}_{n-1}(x)$ for some $j \in \{1, \ldots, m\}$ meaning that we can apply our inductive hypothesis on the derivation of H_j to conclude. □

Lemma 31.
- *Let $\mathbb{C}$ be a set of well originated clauses containing $\mathbb{C}_{std}$. If $\mathbb{C}'$ is the set of clauses obtained from $\mathbb{C}$ by applying the rule Taut, Red, Att, R_ϕ, DataHyp, DataCl, Nat, NatCl, Lem$(\mathcal{L}, \mathcal{S}_p)$ or Ind$(\mathcal{L}_i, \mathcal{S}_p)$ then $\mathbb{C}_{std} \subseteq \mathbb{C}'$.*
- *A clause from $\mathbb{C}_{std}$ cannot be subsumed by a different well-originated, simplified clause.*
- *Let $\mathbb{C}$ be a set of well originated, simplified clauses containing $\mathbb{C}_{std}$. If $\mathbb{C}'$ is the set of clauses obtained from $\mathbb{C}$ by applying the rule GRed$(\mathcal{S}_p)$, then $\mathbb{C}_{std} \subseteq \mathbb{C}'$.*

As a consequence of the first item, if $\mathbb{C}$ is a set of well originated clauses containing $\mathbb{C}_{std}$, then $\mathbb{C}_{std} \subseteq \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ and as consequence of the last two items, if $\mathbb{C}$ is a set of well originated, simplified clauses containing $\mathbb{C}_{std}$, then $\mathbb{C}_{std} \subseteq \mathsf{condense}_{\mathcal{S}_p}(\mathbb{C})$.

Proof. Item 1: Notice that the rules Lem$(\mathcal{L}, \mathcal{S}_p)$ and Ind$(\mathcal{L}_i, \mathcal{S}_p)$ specifically prohibit their application on clauses from $\mathbb{C}_{std}$ hence the result directly holds for these two rules. The rule Taut cannot be applied since hypotheses of clauses in $\mathbb{C}_{std}$ are different from their conclusion. Rules R_ϕ and Nat cannot be applied on $\mathbb{C}_{std}$ as they do not contain any formula. The rule DataCl explicitly prohibits its application on clauses (Rf_g) with $g \in \mathcal{F}_{data}$. The rule NatCl also cannot be applied on R when $g \in \{succ; zero\}$ by definition of the rule. The rule DataHyp explicitly prohibits its application on clauses ($\mathrm{Rf}_{\pi^g_i}$) for all $g \in \mathcal{F}_{data}$ and i. Finally, rules Red and Att cannot be applied on $\mathbb{C}_{std}$ as the hypotheses of these rules cannot be met on these clauses. We therefore conclude that $\mathbb{C}_{std} \subseteq \mathbb{C}'$.

Item 2: Let us now show that a clause from $\mathbb{C}_{std}$ cannot be subsumed by a different well-originated, simplified clause. Consider a clause (Rf_g) with $g \in \mathcal{F}_{data}$, i.e. a clause $R = \mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_n) \to \mathrm{att}_i(g(x_1, \ldots, x_n))$. Let $R' = (H \wedge \phi \to C)$ be a well-originated, simplified clause such that $R' \sqsupseteq R$ and $R \neq R'$. By definition, we deduce that there exists σ such that :

- $C\sigma = \mathrm{att}_i(g(x_1, \ldots, x_n))$
- $H\sigma \subseteq_m \{\!\{\mathrm{att}_i(x_1); \ldots; \mathrm{att}_i(x_n)\}\!\}$
- $\vdash_o \phi\sigma$

$H\sigma \subseteq_m \{\!\{\mathrm{att}_i(x_1); \ldots; \mathrm{att}_i(x_n)\}\!\}$ implies that H is of the form $\mathrm{att}_i(y_1) \wedge \ldots \wedge \mathrm{att}_i(y_k)$ with $k \leq n$. Moreover, w.l.o.g. we have $y_j\sigma = x_j$ for all $j \in \{1, \ldots, k\}$. Since R' is simplified, the

rule DataCl leaves R' unchanged. Hence, we deduce that $C = \mathrm{att}_i(y)$ for some variable y and $y\sigma = g(x_1, \ldots, x_n)$. However, R' is well originated which would imply that $y = y_j$ for some $j \in \{1, \ldots, k\}$. We have a contradiction since $y\sigma = f(x_1, \ldots, x_n)$ and $y_j\sigma = x_j$.

Consider now a clause $(\mathrm{Rf}_{\pi_i^g})$ with $g \in \mathcal{F}_{data}$ and i integer, i.e. a clause $R = \mathrm{att}_n(g(x_1, \ldots, x_n)) \rightarrow \mathrm{att}_n(x_i)$. Let $R' = (H \wedge \phi \rightarrow C)$ be a well-originated, simplified clause such that $R' \sqsupseteq R$ and $R \neq R'$. By definition, there exists σ such that $C\sigma = \mathrm{att}_n(x_i)$ and $H\sigma \subseteq_m \{\!\{\mathrm{att}_n(g(x_1, \ldots, x_n))\}\!\}$. Since $C\sigma = \mathrm{att}_n(x_i)$, we deduce that $C = \mathrm{att}_n(y)$ with $y\sigma = x_i$. Moreover, R' being well-originated and $H\sigma \subseteq_m \{\!\{\mathrm{att}_n(g(x_1, \ldots, x_n))\}\!\}$, we deduce that $H = \mathrm{att}_n(u)$ with $y \in fv(u)$ and $u\sigma = g(x_1, \ldots, x_n)$. However, R' is simplified, so the rule DataHyp leaves R' unchanged. Hence, with $u\sigma = g(x_1, \ldots, x_n)$ we deduce that u must be a variable; and so $y \in fv(u)$ implies $u = y$. We have a contradiction since $u\sigma = g(x_1, \ldots, x_n)$ and $y\sigma = x_i$.

Consider now a clause (Rap), i.e. a clause $R = \mathrm{att}_n(x) \rightarrow \mathrm{att}_{n+1}(x)$. Let $R' = (H \wedge \phi \rightarrow C)$ be a well-originated, simplified clause such that $R' \sqsupseteq R$ and $R \neq R'$. By definition, there exists σ such that $C\sigma = \mathrm{att}_{n+1}(x)$ and $H\sigma \subseteq_m \{\!\{\mathrm{att}_n(x)\}\!\}$. Hence $C = \mathrm{att}_{n+1}(y)$. Moreover, R' being well originated, we deduce that $H = \mathrm{att}_n(y)$ which implies that $R = R'$ and so we obtain a contradiction (since we assume $R \neq R'$).

Finally, consider a clause (Rl), i.e. a clause $R = \mathrm{msg}_n(x, y) \wedge \mathrm{att}_n(x) \rightarrow \mathrm{att}_n(y)$. Let $R' = (H \wedge \phi \rightarrow C)$ be a well-originated, simplified clause such that $R' \sqsupseteq R$ and $R \neq R'$. Therefore, we deduce the existence of a substitution σ where R' can only be one of the following clauses:

- $\mathrm{att}_n(x') \rightarrow \mathrm{att}_n(y')$ with $x'\sigma = x$ and $y'\sigma = y$
- $\mathrm{msg}_n(x', y') \rightarrow \mathrm{att}_n(z')$ with $x'\sigma = x$, $y'\sigma = y$ and $z'\sigma = y$
- $\rightarrow \mathrm{att}_n(y')$ with $y'\sigma = y$

None of these clauses are well originated (even in the second case when taking $y' = z'$) which is in contradiction with our hypothesis on R'.

Item 3: Let us prove that the rule $\mathrm{GRed}(\mathcal{S}_p)$ cannot be applied on clauses from $\mathbb{C}_{std}$. Consider a selection free clause $R = (H \rightarrow F)$ from $\mathbb{C}_{std}$. Moreover, consider a partial derivation $\mathcal{D}$ of F from the selection free clauses of $\mathbb{C} \setminus \{R\}$. For the rule $\mathrm{GRed}(\mathcal{S}_p)$ to be applied, we also require that $\mathbb{F}_{us}(\mathcal{D}) \rightarrow F \sqsupseteq H \rightarrow F$. We do a case analysis on the clause R.

Case R being a clause (Rf_g) with $g \in \mathcal{F}_{data}$, i.e. a clause $R = \mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_n) \rightarrow \mathrm{att}_i(g(x_1, \ldots, x_n))$: Consider the child η of the root of $\mathcal{D}$ and let us denote by $R' = (H \rightarrow C)$ the clause labeling η. By definition of a partial derivation, if η has m outgoing edges labeled by $H_1, \ldots, H_m$ then $R' \sqsupseteq H_1 \wedge \ldots \wedge H_m \rightarrow \mathrm{att}_i(g(x_1, \ldots, x_n))$. Hence there exists σ such that $C\sigma = \mathrm{att}_i(g(x_1, \ldots, x_n))$. Note that R' is simplified, i.e. the rule DataCl cannot be applied on it. Hence we deduce that $C = \mathrm{att}_i(y)$ for some variable y such that $y\sigma = g(x_1, \ldots, x_n)$. Since $\mathrm{att}_i(y)$ is an unselectable fact and R' is selection free, we deduce from Definition 19 that all facts in H are unselectable. Moreover, with R' being well-originated, we obtain that $\mathrm{att}_j(y) \in H$ for some $j \leq i$. Since the rule Taut cannot be applied on R', we deduce that $j < i$. By applying Lemma 30, we deduce that there exist j' and y' such that $\mathrm{att}_{j'}(y') \in \mathbb{F}_{us}(\mathcal{D})$ with $j' < i$. However $\mathbb{F}_{us}(\mathcal{D}) \rightarrow F \sqsupseteq H \rightarrow F$, so the phase of all facts in $\mathbb{F}_{us}(\mathcal{D})$ is i, which leads to a contradiction.

Case R being a clause ($\mathrm{Rf}_{\pi_i^g}$) with $g \in \mathcal{F}_{data}$ and i integer: By definition, the clause is not selection free so the rule $\mathrm{GRed}(\mathcal{S}_p)$ cannot be applied on it.

Case R being a clause (Rap), i.e. $R = \mathrm{att}_n(x) \to \mathrm{att}_{n+1}(x)$. As showed in Lemma 30, the partial derivation must contain a branch of sequential applications of rules (Rap). In our case, it would imply that the clause R was used in the partial derivation $\mathcal{D}$ which contradicts our hypotheses.

Case R being a clause (Rl), i.e. a clause $R = \mathrm{msg}_n(x, y) \wedge \mathrm{att}_n(x) \to \mathrm{att}_n(y)$. The clause R is not selection-free, so $\mathrm{GRed}(\mathcal{S}_p)$ cannot be applied on it, which allows us to conclude. □

Preamble to lemmas. We now focus on the core properties satisfied by our transformation rules on clauses. To avoid repeating the same hypothesis in the following lemmas, we consider the following preamble for all the lemmas in this section:

> *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of* ProVerif *IO-n_{IO}-compliant lemmas. Let $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i)$ and $\mathcal{M}$ be a multiset such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \mathcal{M})$ holds.*

Notice that this preamble corresponds to the hypotheses of Theorem 2.

C.2 Soundness of $\mathsf{simplify}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R\})$

Lemma 32. *Let $\mathbb{C}$ be a well-originated set of clauses containing $\mathbb{C}_{std}$ and $\mathbb{C}_e(T)$. Let R be a clause. Let $\mathcal{D}_{F_0} = \mathcal{D}[\mathcal{D}_R]$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \{R\}$ such that $\{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$ and nodes in $\mathcal{D}_R$ are labeled by clauses from $\mathbb{C}$ except the root that is labeled by R and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$.*

There exist two derivation contexts $\mathcal{D}_\pi[\_]$, $\mathcal{D}'[\_]$ and a derivation $\mathcal{D}'_R$ such that:

1. $\mathcal{D}[\_] = \mathcal{D}'[\mathcal{D}_\pi[\_]]$
2. $\mathcal{D}_\pi[\_]$ *is a context derivation with a unique hole and whose nodes are labeled by a clause* $\mathrm{att}_n(f(x_1, \ldots, x_m)) \to \mathrm{att}_n(x_i)$ *for some* $f \in \mathcal{F}_{data}$ *and* $i \in \{1, \ldots, m\}$
3. $\mathcal{D}'_R$ *is a derivation from* $\mathbb{C} \cup \mathsf{simplify}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R\})$ *such that all nodes except the root are labeled by clauses from* $\mathbb{C}$.
4. $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$
5. $|\mathcal{D}'_R| \leq |\mathcal{D}_\pi[\mathcal{D}_R]|$
6. *if $\mathcal{D}_R$ is strictly selection free then $\mathcal{D}'_R$ is strictly selection free.*

Proof. By definition, $\mathsf{simplify}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R\})$ is the repeated application on $\{R\}$ of the rules Taut, Red, Att, R_ϕ, DataHyp, DataCl, Nat, NatCl, $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$ and $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$. Thus, the computation of $\mathsf{simplify}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R\})$ can be seen as sequence $\mathbb{C}_0, \ldots, \mathbb{C}_N$ where $\mathbb{C}_0 = \{R\}$, $\mathbb{C}_N = \mathsf{simplify}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R\})$ and for all $i \in \{1, \ldots, N\}$, $\mathbb{C}_i$ is obtained from $\mathbb{C}_{i-1}$ by application of one the rules Taut, Red, Att, R_ϕ, DataHyp, DataCl, Nat, NatCl, $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$ and $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$.

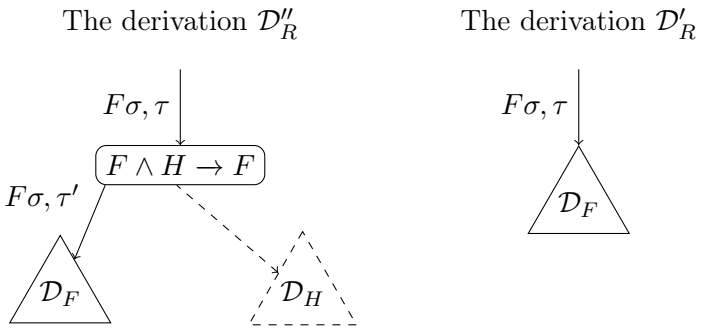
The proof of the lemma is thus done by induction on N by showing that the properties stated in the lemma hold for all $i \in \{0, \ldots, N\}$. In particular, for item 3, we prove that $\mathcal{D}'_R$ is a derivation from $\mathbb{C} \cup \mathbb{C}_i$ such that all nodes except the root are labeled by clauses from $\mathbb{C}$.

Note that by Lemma 31, we know that $\mathbb{C}_i$ contains $\mathbb{C}_{std}$ for all $i \in \{1, \ldots, N\}$ since $\mathbb{C}$ contains $\mathbb{C}_{std}$.

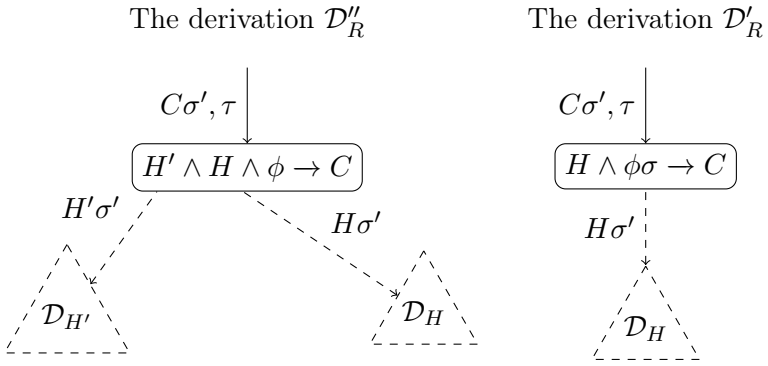
Base case $i = 0$: Since $\mathbb{C}_0 = \{R\}$, the result directly holds by taking $\mathcal{D}_\pi$ the derivation context containing just a hole (no other node) and $\mathcal{D}'_R = \mathcal{D}_R$.

Inductive step $i > 0$: In such a case, we apply our inductive hypothesis on $i-1$ which gives us the existence of $\mathcal{D}''_\pi[\_]$, $\mathcal{D}''[\_]$ and $\mathcal{D}''_R$ satisfying the inductive properties. In particular, $\mathcal{D}''_R$ is a derivation from $\mathbb{C} \cup \mathbb{C}_{i-1}$ such that all nodes except the root are labeled by clauses from $\mathbb{C}$. Let us look at the transformation applied on $\mathbb{C}_{i-1}$ to obtain $\mathbb{C}_i$. Notice that all the transformation rules only affect one clause at a time. Thus, there exists R' such that $\mathbb{C}_{i-1} = \mathbb{C}'_{i-1} \cup \{R'\}$ and $\mathbb{C}_i = \mathbb{C}'_{i-1} \cup \mathbb{C}'$ where $\mathbb{C}'$ is the result of the application of a transformation rule on R'. Hence, if the root of $\mathcal{D}''_R$ is labeled by a clause of $\mathbb{C}'_{i-1}$ or $\mathbb{C}$, the result trivially holds by taking $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$, $\mathcal{D}'[\_] = \mathcal{D}''[\_]$ and $\mathcal{D}_R = \mathcal{D}'_R$. Otherwise, the root of $\mathcal{D}''_R$ is labeled by R'. Note that by definition of the transformation rules, R' cannot be a clause (Rf) for a data constructor function symbol or one of its projections. Thus, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$ implies that all derivations $\mathcal{D}'''$ directly outgoing R' satisfy $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'''$. We do a case analysis on the transformation rule applied on R':

- Rule Taut: In such a case R' is of the form $F \wedge H \to F$ and $\mathcal{D}''_R$ is some derivation of $F\sigma$ at some step τ for some substitution σ. Since F is also in the hypothesis of R', there exists a derivation $\mathcal{D}_F$ of $F\sigma$ at some step τ'. Note that since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$, we deduce that $\tau' \leq \tau$. Thus, we can take for $\mathcal{D}'_R$ the derivation $\mathcal{D}_F$ with $F\sigma, \tau$ as incoming edge, $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$ and $\mathcal{D}'[\_] = \mathcal{D}''[\_]$. Since we removed a node, we directly have that $|\mathcal{D}'_R| \leq |\mathcal{D}''_R|$ and so $|\mathcal{D}'_R| \leq |\mathcal{D}_\pi[\mathcal{D}_R]|$. Moreover $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_F$ and since $\tau' \leq \tau$, we have $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$. Finally, if $\mathcal{D}_R$ is strictly selection free then $\mathcal{D}''_R$ is strictly selection free by our inductive hypothesis which implies that $\mathcal{D}_F$ is selection free which implies that $\mathcal{D}_F$ is strictly selection free.

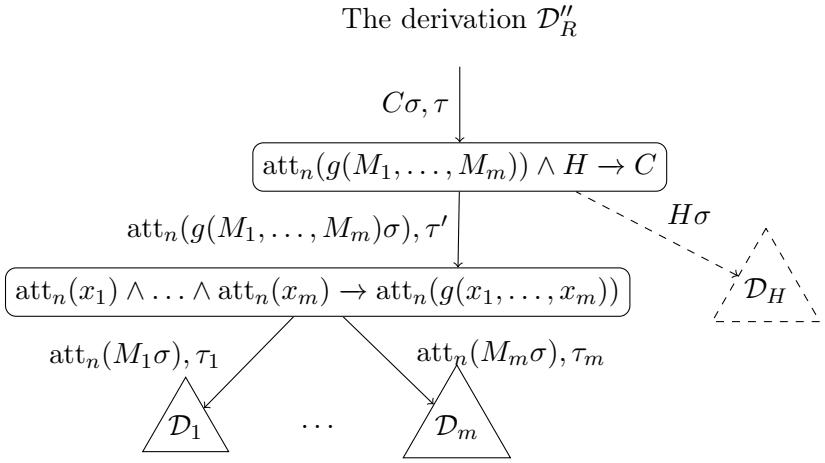


- Rule Red: In such a case, R' is of the form $H' \wedge H \wedge \phi \to C$ such that $H'\sigma \subseteq H$, $\phi \models \phi\sigma$ and $dom(\sigma) \cap fv(H, C) = \emptyset$. Moreover, by Definition 17 of a derivation, there exist conjunctions of derivations $\mathcal{D}_H, \mathcal{D}'_H$ and σ' such that $\mathcal{D}''_R$ is some derivation of $C\sigma'$ at some step τ, $\vdash_i \phi\sigma'$ and $\mathcal{D}_H$ and $\mathcal{D}_{H'}$ are the conjunction derivations of $H\sigma'$ and $H'\sigma'$ respectively. We build the derivation $\mathcal{D}'_R$ by replacing the label of the root with $H \wedge \phi\sigma \to C$ and by only keeping $\mathcal{D}_H$ as outgoing derivation. Note that for $\mathcal{D}'_R$ to be a derivation, we need to show that $\vdash_i \phi\sigma\sigma'$. By hypothesis, $\phi \models \phi\sigma$ hence $\vdash_i \phi\sigma'$ implies $\vdash_i \phi\sigma\sigma'$. We conclude by taking $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$ and $\mathcal{D}'[\_] = \mathcal{D}''[\_]$. The desired properties on $\mathcal{D}_\pi[\_], \mathcal{D}'[\_]$ and $\mathcal{D}'_R$ are a direct consequence of the inductive hypothesis.



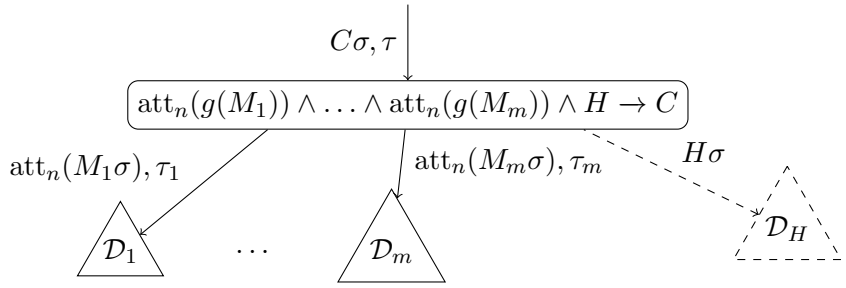
- Rule Att: In such a case, R' is of the form $\mathrm{att}_i(x) \wedge H \wedge \phi \to C$ where x does not appear in H, C. This case is similar to the previous one since we only remove the fact $\mathrm{att}_i(x)$ from the clause.

- Rule DataHyp: In such a case, R' is of the form $\mathrm{att}_n(g(M_1, \ldots, M_m)) \wedge H \to C$ where $g \in \mathcal{F}_{data}$ and $R' \neq (\mathrm{Rf}_{\pi_i^g})$ for all i. Moreover, $\mathcal{D}''_R$ is the derivation of $C\sigma$ at step τ for some substitution σ and some step τ. Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$ and in particular item 4, we deduce that the derivation for $\mathrm{att}_n(g(M_1, \ldots, M_m))\sigma$ is rooted by a node labeled with the clause $\mathrm{att}_n(x_1) \wedge \ldots \wedge \mathrm{att}_n(x_m) \to \mathrm{att}_n(g(x_1, \ldots, x_m))$. Hence $\mathcal{D}''_R$ is of the following form:



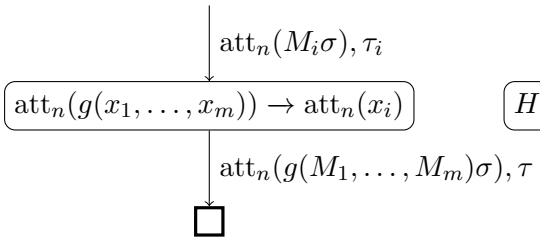
Thanks to $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$, we also know that $\tau_i \leq \tau'$ for all $i \in \{1, \ldots, m\}$ and $\tau' \leq \tau$ (note that if $\mathrm{att}_n \in \mathcal{S}_p$ then these inequalities are strict). By definition of the transformation rule DataHyp, the clause R' is replaced by $\mathrm{att}_n(M_1) \wedge \ldots \wedge \mathrm{att}_n(M_m) \wedge H \to C$. Since $\tau_i \leq \tau$ (or $\tau_i < \tau$ when $\mathrm{att}_n \in \mathcal{S}_p$) and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$ for all $i \in \{1, \ldots, m\}$, we can build the derivation $\mathcal{D}'_R$ by directly using $\mathcal{D}_1, \ldots, \mathcal{D}_n$ as outgoing derivations for the clause $\mathrm{att}_n(M_1) \wedge \ldots \wedge \mathrm{att}_n(M_m) \wedge H \to C$. We conclude by taking $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$ and $\mathcal{D}'[\_] = \mathcal{D}''[\_]$.

The derivation $\mathcal{D}'_R$

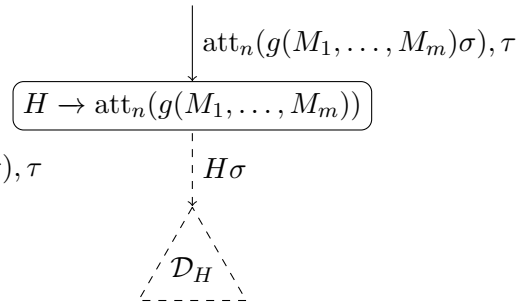


- Rule DataCl: In such a case, R' is of the form $H \to \mathrm{att}_n(g(M_1, \ldots, M_m))$ with $g \in \mathcal{F}_{data}$ and $R' \neq (\mathrm{Rf}_g)$. The transformation rule replaces R' with the set of clauses $\{H \to \mathrm{att}_n(M_i)\}_{i=1}^m$. Moreover, $\mathcal{D}''_R$ is the derivation of $\mathrm{att}_n(g(M_1, \ldots, M_m)\sigma)$ at step τ for some substitution σ and some step τ. Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$ and in particular item 4, we deduce that the node connected to the hole in $\mathcal{D}''[\_]$ is labeled with the clause $\mathrm{att}_n(g(x_1, \ldots, x_m)) \to \mathrm{att}_n(x_i)$ for some $i \in \{1, \ldots, m\}$. Therefore, we can split $\mathcal{D}''[\_]$ into two derivation contexts $\mathcal{D}''_1[\_]$ and $\mathcal{D}_{\pi_i^g}[\_]$ such that $\mathcal{D}''[\_] = \mathcal{D}''_1[\mathcal{D}_{\pi_i^g}[\_]]$ and $\mathcal{D}_{\pi_i^g}[\_]$ contains a single node labeled with $\mathrm{att}_n(g(x_1, \ldots, x_m)) \to \mathrm{att}_n(x_i)$.

The derivation context $\mathcal{D}_{\pi_i^g}[\_]$

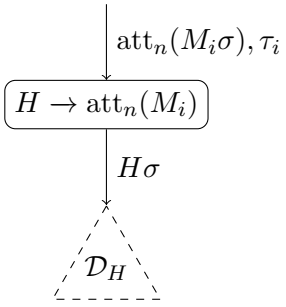


The derivation $\mathcal{D}''_R$



Since the transformation rule replaces R' with the set of clauses $\{H \to \mathrm{att}_n(M_i)\}_{i=1}^m$, we can build the $\mathcal{D}'_R$ of $\mathrm{att}_n(M_i\sigma)$ at step τ_i by using the clause $H \to \mathrm{att}_n(M_i)$. Notice that contrary to the previous transformation rule, $\mathcal{D}'_R$ and $\mathcal{D}''_R$ are not derivations of the same fact.

The derivation $\mathcal{D}'_R$



Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$, we deduce that $\tau \leq \tau_i$ (the inequality is strict when $\mathrm{att}_n \in \mathcal{S}_p$). Hence, we obtain that $\mathcal{D}''_1[\mathcal{D}'_R]$ is a derivation of F_0 such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''_1[\mathcal{D}'_R]$. Since $\mathcal{D}_{\pi_i^g}[\_]$ contains only a projection clause of a data constructor symbol, we can define $\mathcal{D}_\pi = \mathcal{D}_{\pi_i^g}[D''_\pi[\_]]$ and $\mathcal{D}'[\_] = \mathcal{D}''_1[\_]$ to obtain $\mathcal{D}[\_] = \mathcal{D}'[\mathcal{D}_\pi[\_]]$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$. By our inductive hypothesis, $|\mathcal{D}''_R| \leq |\mathcal{D}''_\pi[\mathcal{D}_R]|$. But $|\mathcal{D}_{\pi_i^g}[\mathcal{D}''_R]| = |\mathcal{D}_H| + 2$ and $|\mathcal{D}'_R| \leq |\mathcal{D}_H| + 1$ (the inequality may be strict if $\mathcal{D}'_R$ becomes the basic derivation of a natural number). Thus, $|\mathcal{D}'_R| \leq |\mathcal{D}_{\pi_i^g}[\mathcal{D}''_R]|$ and so $|\mathcal{D}'_R| \leq |\mathcal{D}_\pi[\mathcal{D}_R]|$.

- Rule Nat: If R' is of the form $H \wedge \phi \to C$ then by definition of the substitution, $\mathcal{D}''_R$ is the derivation of the fact $C\sigma$ for some substitution σ and $\vdash_i \phi\sigma$. One can easily see that the transformations applied in Nat either modify the clause R' by replacing ϕ with an equivalent constraint formula ϕ', yielding the clause $R'' = H \wedge \phi' \to C$, or removes R' when ϕ is unsatisfiable. In the latter case, $\vdash_i \phi\sigma$ in fact implies that the rule Nat does not remove R' since ϕ is satisfiable. There, we conclude by taking $\mathcal{D}'[\_] = \mathcal{D}''[\_]$, $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$ and $\mathcal{D}'_R$ the derivation obtained from $\mathcal{D}''_R$ by replacing the label of the root with R''.

- Rule NatCl: In such a case, R' is of the form $H \wedge \phi \to \mathrm{att}_n(M)$ with $\phi \models nat(M)$ and $R' \notin \{(\mathrm{R}+), (\mathrm{R}0)\}$. Moreover, $\mathcal{D}''_R$ is the derivation of $\mathrm{att}_n(M\sigma)$ at step τ for some substitution σ and some step τ. Since $\phi \models nat(M)$, we deduce that $M\sigma$ is a natural number. By hypothesis, $\mathbb{C}'$ contains the clauses $\mathbb{C}_{std}$ and in particular the clauses $\to \mathrm{att}_n(zero)$ and $\mathrm{att}_n(x) \to \mathrm{att}_n(succ(x))$. We build $\mathcal{D}'_R$ depending on whether $\mathrm{att}_n \in \mathcal{S}_p$ or not.

  - If $\mathrm{att}_n \in \mathcal{S}_p$ then $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$ implies that either M is a variable and $T, \tau \vdash \mathrm{att}_n(M\sigma)$ or M is not a variable and so $M\sigma \in \mathcal{A}(T[\tau])$ (see item 4 of Definition 18). In both cases, we deduce that $M\sigma \in \mathcal{A}(T[\tau])$. But $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i)$ meaning that T is data compliant. In particular, $T[0] \xrightarrow{\text{I-App}(0)}_i \overset{1}{\underset{\mathsf{r}}{\Rightarrow}} \ldots \overset{M\sigma}{\underset{\mathsf{r}}{\Longrightarrow}} T[M\sigma + 1]$ (where we see $M\sigma$ here as an actual natural number in $\mathbb{N}$). Note that $M\sigma + 1$ is thus the smallest step in which $M\sigma$ occurs in the attacker knowledge. Therefore $M\sigma + 1 \leq \tau$ meaning that we can take $\mathcal{D}'_R$ as the basic derivation of $M\sigma$ such that the incoming edge of the root is labeled by $\mathrm{att}_n(M\sigma), \tau$ and for all other edges, if the edge is labeled $\mathrm{att}_n(k), \tau'$ then $\tau' = k + 1$. Since $M\sigma + 1 \leq \tau$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_R$. By taking $\mathcal{D}'[\_] = \mathcal{D}''[\_]$ and $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$, we obtain $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$. By definition, $|\mathcal{D}'_R| = 0$ hence we trivially have that $|\mathcal{D}'_R| \leq |\mathcal{D}_\pi[\mathcal{D}_R]|$. Finally, since $\mathbb{C}'$ contains the clauses $\to \mathrm{att}_n(zero)$ and $\mathrm{att}_n(x) \to \mathrm{att}_n(succ(x))$, we conclude that $\mathcal{D}'_R$ satisfy the item 3 of the desired properties, which allows us to conclude.
  - If $\mathrm{att}_n \notin \mathcal{S}_p$ then we build $\mathcal{D}'_R$ as being the basic derivation $M\sigma$ such that the incoming edge of the root is labeled $\mathrm{att}_n(M\sigma), \tau$ and the step of all other edges is also τ. Note that having τ being the step of all edges still allows us to obtain $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_R$ since $\mathrm{att}_n \notin \mathcal{S}_p$. We conclude by taking $\mathcal{D}'[\_] = \mathcal{D}''[\_]$ and $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$.

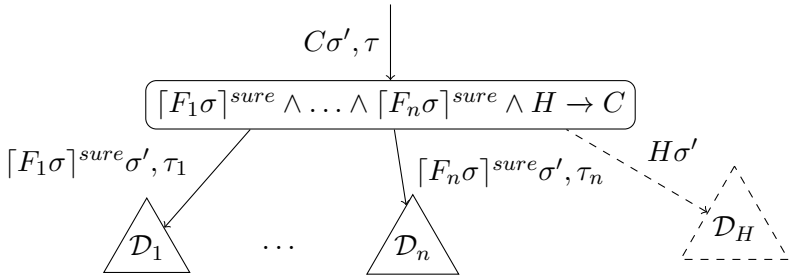
- Rule $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$ and $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$: We prove these two rules at the same time. We distinguish two cases. Case 1 is when $\mathrm{Ind}(\mathcal{L}_i, \mathcal{S}_p)$ is applied or $\mathrm{Lem}(\mathcal{L}, \mathcal{S}_p)$ is applied and

all facts F_i are matched with hypotheses of R'. Case 2 is when $\text{Lem}(\mathcal{L}, \mathcal{S}_p)$ is applied and a fact F_i is matched with the conclusion of R'.

In case 1, there exist a substitution σ and $\psi = (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \phi_j) \in \mathcal{L} \cup \mathcal{L}_i$ such that $R' = (\lceil F_1\sigma \rceil^{sure} \wedge \ldots \lceil F_n\sigma \rceil^{sure} \wedge H \rightarrow C)$ and for all $i \in \{1, \ldots, n\}$, $pred(F_i) \in \mathcal{S}_p$.

Thus, in case 1, $\mathcal{D}''_R$ is the derivation of $C\sigma'$ at step some τ for some σ' as presented below:

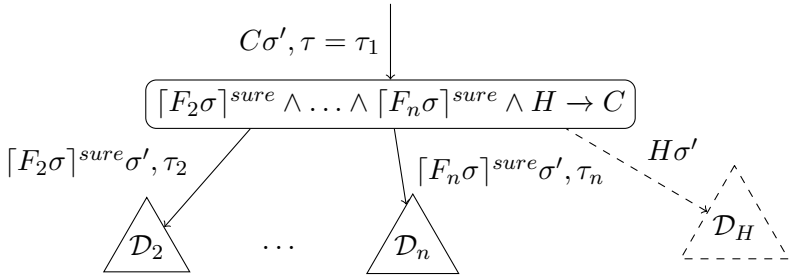
The derivation $\mathcal{D}''_R$ in Case 1



In case 2, w.l.o.g. there exist a substitution σ and $\psi = (\bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m} \phi_j) \in \mathcal{L}$ such that $R' = (\lceil F_2\sigma \rceil^{sure} \wedge \ldots \lceil F_n\sigma \rceil^{sure} \wedge H \rightarrow C)$, $\lceil F_1\sigma \rceil^{may} = C$ and for all $i \in \{1, \ldots, n\}$, $pred(F_i) \in \mathcal{S}_p$.

By denoting $\tau_1 = \tau$, we obtain that in case 2, $\mathcal{D}''_R$ is the derivation of $C\sigma'$ at step some τ for some σ' as presented below:

The derivation $\mathcal{D}''_R$ in Case 2



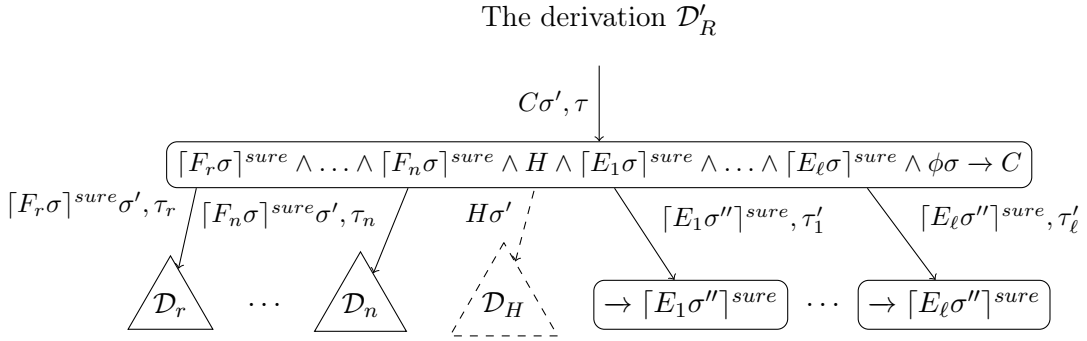
Recall that $R' \notin \mathbb{C}_{std}$ by definition of the rules. Furthermore, also recall for all events E and steps τ', $T, \tau' \vdash_{IO}^{n_{IO}} E$ if and only if $T, \tau' \vdash_{IO}^{n_{IO}} \lceil E \rceil^{sure}$ if and only if $T, \tau' \vdash_{IO}^{n_{IO}} \lceil E \rceil^{may}$. Recall that $\mathcal{D}''[\mathcal{D}''_R]$ is a derivation of some fact F_0 at step τ_0 with $\{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$. Hence since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$ and for all $i \in \{1, \ldots, n\}$, $pred(F_i) \in \mathcal{S}_p$, we deduce that for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash_{IO}^{n_{IO}} \lceil F_i\sigma \rceil^{sure}\sigma'$. Moreover, in case 1, we have $\tau_i < \tau \leq \tau_0$ for all $i \in \{1, \ldots, n\}$, meaning that $\{\!\{\tau_1, \ldots, \tau_n\}\!\} <_m \{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$. Thus, in both cases 1 and 2, we can apply Lemma 29 which allows us to deduce that there exist $j \in \{1, \ldots, m\}$, a substitution σ'', $\tau'_1, \ldots, \tau'_j$ steps some events $E_1, \ldots, E_\ell$ and a formula ϕ' such that

- $\phi_j = E_1 \wedge \ldots \wedge E_\ell \wedge \phi'$
- for all $i \in \{1, \ldots, n\}$, $F_i\sigma\sigma' = F_i\sigma''$
- for all $k \in \{1, \ldots, \ell\}$, $\tau'_k \leq \max(\tau_1, \ldots, \tau_n)$ and $T, \tau'_k \vdash_{IO}^{n_{IO}} E_k\sigma''$

$- \vdash_{IO}^{n_{IO}} \phi'\sigma''$

In Case 1, we already showed that $\tau_i < \tau \leq \tau_0$ for all $i \in \{1, \ldots, n\}$. Hence, $\max(\tau_1, \ldots, \tau_n) < \tau$ and so for all $k \in \{1, \ldots, \ell\}$, $\tau'_k < \tau$. In Case 2, we do not have $\max(\tau_1, \ldots, \tau_n) < \tau$ since $\tau_1 = \tau$. However, we know by hypothesis that for all $k \in \{1, \ldots, \ell\}$, $mgu(E_k\sigma, C) = \bot$. Thus $E_k\sigma\sigma'$ and $C\sigma'$ are not unifiable and since $F_i\sigma\sigma' = F_i\sigma''$ for all $i \in \{1, \ldots, n\}$, we conclude that $E_k\sigma'' \neq C\sigma'$. But we already showed that $T, \tau'_k \vdash_{IO}^{n_{IO}} E_k\sigma''$ and $T, \tau \vdash_{IO}^{n_{IO}} C\sigma'$. Since $E_k\sigma''$ is an event, we obtain that $\tau'_k \neq \tau$. With $\tau'_k \leq \max(\tau_1, \ldots, \tau_n)$ and $\tau_1 = \tau$, we conclude that $\tau'_k < \tau$.

We can now build the derivation $\mathcal{D}'_R$ by replacing the clause R' with the clause $\lceil F_r\sigma \rceil^{sure} \wedge \ldots \lceil F_n\sigma \rceil^{sure} \wedge H \wedge \lceil E_1\sigma \rceil^{sure} \wedge \ldots \wedge \lceil E_\ell\sigma \rceil^{sure} \wedge \phi\sigma \rightarrow C$ and $r = 1$ (resp. 2) in Case 1 (resp. 2) as follows:



Let us take $\mathcal{D}_\pi[\_] = \mathcal{D}''_\pi[\_]$ and $\mathcal{D}'[\_] = \mathcal{D}''[\_]$. Since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}''[\mathcal{D}''_R]$, the derivation $\mathcal{D}'_R$ adding only subderivations containing only events and the fact that $\tau'_1, \ldots, \tau'_\ell < \tau$, we deduce that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$. Moreover, the size of a derivation does not count the nodes labeled by an event clause (i.e. $\rightarrow$ s-event(o, ev)) hence $|\mathcal{D}'_R| = |\mathcal{D}''_R|$. With our inductive hypothesis, we obtain $|\mathcal{D}'_R| \leq |\mathcal{D}_\pi[\mathcal{D}_R]|$. Finally, by definition of the selection function, events are unselectable facts. Hence if $\mathcal{D}''_R$ is strictly selection free then so is $\mathcal{D}'_R$. This allows us to conclude. □

Corollary 2. *Let $\mathbb{C}$ be a well-originated set of clauses containing $\mathbb{C}_{std}$. Let R be a clause. Let $\mathcal{D}_{F_0}$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \{R\} \cup \mathbb{C}_e(T)$ such that $\{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$. There exists a derivation $\mathcal{D}'_{F_0}$ of F_0 at step τ_0 from $\mathbb{C} \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_{F_0}$.*

Proof. Note that when $R \in \mathbb{C}$ or $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}) = \{R\}$, the result trivially holds. Otherwise, notice that $\mathcal{D}_{F_0}$ that is also a derivation from $\mathbb{C} \cup \{R\} \cup \mathbb{C}_e(T) \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$. We prove the result by induction on the number N of nodes in $\mathcal{D}_{F_0}$ labeled with the clause R. When $N = 0$ (the base case), $\mathcal{D}_{F_0}$ is a derivation from $\mathbb{C} \cup \mathbb{C}_e(T) \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ hence the result holds with $\mathcal{D}'_{F_0} = \mathcal{D}_{F_0}$. When $N > 0$ (the inductive step), we can take the smallest subderivation of $\mathcal{D}_{F_0}$ that has R as root, i.e. $\mathcal{D}_{F_0} = \mathcal{D}[\mathcal{D}_R]$ where the root of $\mathcal{D}_R$ is labeled by R and all other nodes in $\mathcal{D}_R$ are labeled by clauses of $\mathbb{C} \cup \mathbb{C}_e(T) \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$. Hence we can apply Lemma 32 with $\mathbb{C}_0 = \mathbb{C} \cup \mathbb{C}_e(T) \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ to obtain that there exist two derivation contexts $\mathcal{D}_\pi[\_]$, $\mathcal{D}'[\_]$ and a derivation $\mathcal{D}'_R$ such that $D[\_] = \mathcal{D}'[D_\pi[\_]]$ (item 1), $\mathcal{D}'_R$ is a derivation from $\mathbb{C}_0 \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}) = \mathbb{C}_0$ (item 3) and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'[\mathcal{D}'_R]$ (item 4).

Note that $D[\_] = \mathcal{D}'[D_\pi[\_]]$ and $\mathcal{D}_{F_0} = \mathcal{D}[\mathcal{D}_R]$ ensure that $\mathcal{D}'[\mathcal{D}'_R]$ is also a derivation of F_0 at step τ_0. Since $\mathcal{D}_R$ was rooted by R and $R \notin \mathbb{C}_0$, we deduce that $\mathcal{D}'[\mathcal{D}'_R]$ has strictly fewer nodes labeled by R than the derivation $\mathcal{D}_{F_0}$. Hence we conclude by applying the inductive hypothesis on $\mathcal{D}'[\mathcal{D}'_R]$. □

Corollary 3. *Let $\mathbb{C}$ be a well-originated set of clauses containing $\mathbb{C}_{std}$. Let $\mathcal{D}_{F_0}$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $\{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$. There exists a derivation $\mathcal{D}'_{F_0}$ of F_0 at step τ_0 from $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_{F_0}$.*

Proof. Noticing that $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_1, \ldots, R_n\}) = \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_1\}) \cup \ldots \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_n\})$, the result directly follows from Corollary 2. □

C.3 Soundness of $\mathsf{condense}_{\mathcal{S}_p}(\mathbb{C})$

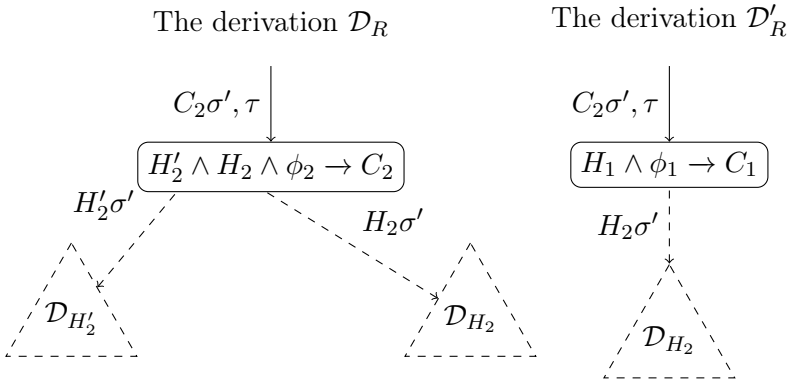
Lemma 33. *Let $\mathbb{C}$ be a well originated set of clauses containing $\mathbb{C}_{std}$. Let R be a well originated clause. Let $\mathcal{D}_{F_0} = \mathcal{D}[\mathcal{D}_R]$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \{R\}$ such that all nodes in $\mathcal{D}_R$ are labeled by clauses from $\mathbb{C}$ except the root that is labeled by R and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$.*

If there exists a well originated clause R' such that $R' \sqsupseteq R$ then there exists $\mathcal{D}'_R$ such that:

1. *$\mathcal{D}'_R$ is a derivation from $\mathbb{C} \cup \{R'\}$ such that all nodes except the root are labeled by clauses from $\mathbb{C}$.*
2. *$T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}'_R]$*
3. *$|\mathcal{D}'_R| \leq |\mathcal{D}_R|$*
4. *if $\mathcal{D}_R$ is strictly selection free then $\mathcal{D}'_R$ is strictly selection free.*

Proof. Assume that $R' \sqsupseteq R$. By definition and by denoting $R = H_2 \wedge H'_2 \wedge \phi_2 \rightarrow C_2$ and $R' = H_1 \wedge \phi_1 \rightarrow C_1$, we deduce that there exists a substitution σ such that $C_1\sigma = C_2$, $H_1\sigma = H_2$ and $\phi_2 \models \phi_1\sigma$. Moreover, since $\mathcal{D}_R$ is a derivation whose root is labeled by R, there exist two conjunctions of derivations $\mathcal{D}_{H_2}$, $\mathcal{D}_{H'_2}$ and a substitution σ' such that $\mathcal{D}_R$ is some derivation of $\mathcal{C}_2\sigma'$ at some step τ, $\vdash_i \phi_2\sigma'$ and $\mathcal{D}_{H_2}$, $\mathcal{D}_{H'_2}$ are the conjunction derivations of $H_2\sigma'$ and $H'_2\sigma'$ respectively.

Note that $C_2\sigma' = C_1\sigma\sigma'$ and $H_2\sigma' = H_1\sigma\sigma'$. Hence, we build the derivation $\mathcal{D}'_R$ by replacing R in R' as root and by only keeping the conjunction derivation $\mathcal{D}_{H_2}$. Note that $\vdash_i \phi_1\sigma\sigma'$ and $\phi_2 \models \phi_1\sigma$ imply $\vdash_i \phi_1\sigma\sigma'$. Hence $\mathcal{D}'_R$ is really a derivation.



Items 1, 3 and 4 of the lemma are directly obtained by construction of $\mathcal{D}'_R$. Thanks to Lemma 31, we know that $R \neq$ (Rl) and R is not the clause (Rf) for a data constructor or one of its projections. We can therefore deduce $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}'_R]$ directly from $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}_R]$ which allows us to conclude. □

Corollary 4. *Let $\mathbb{C} = \mathbb{C}' \cup \{R; R'\}$ be a well-originated set of clauses containing $\mathbb{C}_{std}$ such that $R' \sqsupseteq R$. Let $\mathcal{D}_{F_0}$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C}$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$. There exists a derivation $\mathcal{D}'_{F_0}$ of F_0 at step τ_0 from $\mathbb{C}' \cup \{R'\}$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_{F_0}$.*

Proof. The proof is done by induction on the number of nodes labeled by R, the inductive step being a direct application of Lemma 33. Note that by Lemma 31, the clause in $\mathbb{C}_{std}$ are also contained in the set $\mathbb{C}' \cup \{R'\}$. □

Lemma 34. *Let $\mathbb{C}$ be a set of well originated, simplified clauses containing $\mathbb{C}_{std}$. Let F be a fact (not necessarily closed) and τ a step. Let $\mathcal{D}_p$ be a partial derivation of F from $\{R \in \mathbb{C} \mid \text{sel}(R) = \emptyset\}$ such that $\mathbb{F}_{us}(\mathcal{D}_p) = U_1 \wedge \ldots \wedge U_n \wedge \phi_u$ where $U_1, \ldots, U_n$ are facts and ϕ_u is a formula.*

If there exist a substitution σ, some steps $\tau_1, \ldots, \tau_n$ and some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_n$ such that $\vdash_{IO}^{n_{IO}} \phi_u\sigma$, for all $i \in \{1, \ldots, n\}$,

- *$\mathcal{D}_i$ is a derivation of $U_i\sigma$ at step τ_i from $\mathbb{C}$*
- *if pred$(U_i) \in \mathcal{S}_p$ then $\tau_i < \tau$ else $\tau_i \leq \tau$*
- *$T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$*
- *if pred$(F) \in \mathcal{S}_p$ then $T, \tau \vdash_{IO}^{n_{IO}} F\sigma$*

then there exists a derivation $\mathcal{D}$ of $F\sigma$ at step τ from $\mathbb{C}$ such that:

- *$T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$*
- *if $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are selection free then $\mathcal{D}$ is selection free*

Proof. The first part of the proof consists of transforming the partial derivation $\mathcal{D}_p$ into a context derivation $\mathcal{D}_c[\__1, \ldots, \__n]$ such that:

- a node η of $\mathcal{D}_p$ labeled by a clause R in $\mathcal{D}_p$ with incoming edge labeled U is transformed into a node η' in $\mathcal{D}_c[\__1, \ldots, \__n]$ also labeled by R with incoming edge labeled $U\sigma, \tau$.
- an unlabeled leaf from $\mathcal{D}_p$ whose incoming edge is labeled by a formula is removed in $\mathcal{D}_c[\__1, \ldots, \__n]$ (as well as its incoming edges).
- an unlabeled leaf from $\mathcal{D}_p$ whose incoming edge is labeled by a fact, i.e. U_i for some $i \in \{1, \ldots, n\}$, is replaced in $\mathcal{D}_c[\__1, \ldots, \__n]$ by the hole $\__i$ with $U_i\sigma, \tau_i$ as incoming edge.

With such context $\mathcal{D}_c[\__1, \ldots, \__n]$, we will obtain a derivation $\mathcal{D}' = \mathcal{D}_c[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ of $F\sigma$ at step τ. We will show that this derivation is almost satisfied by the trace T w.r.t. $\mathcal{S}_p$. In particular, it will satisfy all items of Definition 18 other than Item 4. The second part of the proof consists of transforming $\mathcal{D}'$ into a derivation $\mathcal{D}$ that satisfies all items of Definition 18.

First let us show that $\mathcal{D}_c[\__1, \ldots, \__n]$ really corresponds to a context derivation, i.e. all nodes in the context derivation should satisfy item 3 of Definition 17. Consider a node η' of $\mathcal{D}'[\__1, \ldots, \__n]$ labeled by some clause $R = H \wedge \phi \rightarrow C$. By construction, we know that there exists a node η in $\mathcal{D}_p$ labeled by R with incoming edge labeled U' and some outgoing edges labeled $U'_1, \ldots, U'_m$ and some formula ϕ' (w.l.o.g. we can always consider that there is at most one outgoing edge of a node labeled by a formula). By definition of a partial derivation, we also know that $R \sqsupseteq U'_1 \wedge \ldots \wedge U'_m \wedge \phi' \rightarrow U'$ thus there exists σ' such that $C\sigma' = U'$, $H\sigma' \subseteq_m U'_1 \wedge \ldots \wedge U'_m$ and $\phi' \models \phi\sigma'$.

Consider the substitution $\sigma'\sigma$. Since $C\sigma' = U'$ and $H\sigma' \subseteq_m U'_1 \wedge \ldots \wedge U'_m$, we directly obtain that $C\sigma'\sigma = U'\sigma$ and $H\sigma'\sigma \subseteq_m U'_1\sigma \wedge \ldots \wedge U'_m\sigma$. Let us show that $\vdash^{n_{IO}}_{IO} \phi\sigma'\sigma$. Recall by definition of a partial derivation that the outgoing edge labeled ϕ' is the incoming edge of an unlabeled leave. Hence, the formula ϕ' is part of $\mathbb{F}_{us}(\mathcal{D}_p)$, meaning that $\phi_u \models \phi'$. By hypothesis, $\vdash^{n_{IO}}_{IO} \phi_u\sigma$ hence $\vdash^{n_{IO}}_{IO} \phi'\sigma$. Since $\phi' \models \phi\sigma'$, we deduce that that $\vdash^{n_{IO}}_{IO} \phi\sigma'\sigma$. This allows us to conclude that $R \sqsupseteq U'_1\sigma\sigma' \wedge \ldots \wedge U'_m\sigma\sigma' \rightarrow U'\sigma\sigma'$.

Since by construction, $U'\sigma\sigma'$ is the label of the incoming edge of η' and $U'_1\sigma\sigma', \ldots, U'_n\sigma\sigma'$ are the labels of its outgoing edges, we conclude that $\mathcal{D}_c[\__1, \ldots, \__n]$ is a context derivation.

Let us now show that the derivation $\mathcal{D}' = \mathcal{D}_c[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ satisfies Items 1 to 3 of Definition 18. Item 3 is trivially satisfied since the clauses have no labels in the saturation procedure. Moreover, all internal nodes η' in $\mathcal{D}_c[\__1, \ldots, \__n]$ are labeled by a selection free clause R''. By definition of the selection function, we know that (Rl) is not selection free and so $R'' \neq$ (Rl). Thus Item 1 holds. Finally, by hypothesis of the lemma, we also know that the incoming edge of η' is also labeled by $U'\sigma\sigma', \tau$ for some U' such that $pred(U') \notin \mathcal{S}_p$ by hypothesis of the lemma. Hence, the derivation $\mathcal{D}' = \mathcal{D}_c[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ satisfies item 2 of Definition 18.

Let us now focus on the second part of the proof consisting of building the derivation $\mathcal{D}$ from $\mathcal{D}'$ such that $\mathcal{D}$ satisfies all items of Definition 18 and in particular Item 4. The only problematic case we need to focus on is when a node η_0 from $\mathcal{D}_c[\__1, \ldots, \__n]$ has an incoming edge labeled by $\mathrm{att}_i(f(M_1, \ldots, M_m))$ with $f \in \mathcal{F}_{data}$. Since the projection clauses $\mathrm{att}_i(f(x_1, \ldots, x_m)) \rightarrow \mathrm{att}_i(x_k)$, for all k, are not selection free, they cannot be used in the partial derivation $\mathcal{D}_p$ and so do not occur in $\mathcal{D}'[\__1, \ldots, \__n]$. Thus to satisfy Item 4 of Definition 18, we need to build $\mathcal{D}$ such that the node η_0 is labeled by the rule (Rf_f). We show how to transform the derivation when one node of $\mathcal{D}'$ does not satisfy this property; when several nodes do not satisfy this property, it suffices to apply this transformation on each of these nodes to obtain $\mathcal{D}$.

Let us consider a node η of the derivation context $\mathcal{D}_c[\__1, \ldots, \__n]$ whose incoming edge is labeled by the fact $\mathrm{att}_i(M)$ and let us denote $\mathcal{D}_\eta$ the subderivation of $\mathcal{D}_c[\mathcal{D}_1, \ldots, \mathcal{D}_n]$ rooted in η. Note that for simplicity, we will assimilate the node η with its corresponding node in the partial derivation $\mathcal{D}_p$.

We want to characterise the derivation $\mathcal{D}_\eta$ by extracting from it the parts where clauses (Rap) and (Rf_g) with $g \in \mathcal{F}_{data}$ are applied. We prove by induction on the size of $\mathcal{D}_\eta$ that there exist $\{i_1, \ldots, i_r\} \subseteq \{1, \ldots, n\}$ and some derivation contexts $\mathcal{D}^{\mathcal{F}_{data}}$, $\mathcal{D}^{Rp}_{i_1}$,..., $\mathcal{D}^{Rp}_{i_r}$ such that:

- $\mathcal{D}_\eta = \mathcal{D}^{\mathcal{F}_{data}}[\mathcal{D}'_1, \ldots, \mathcal{D}'_q, \mathcal{D}^{Rp}_{i_1}[\mathcal{D}_{i_1}], \ldots, \mathcal{D}^{Rp}_{i_r}[\mathcal{D}_{i_r}]]$;
- nodes of $\mathcal{D}^{\mathcal{F}_{data}}$ are only labeled by clauses (Rf_g) for $g \in \mathcal{F}_{data}$;
- nodes of $\mathcal{D}^{Rp}_{i_1}, \ldots, \mathcal{D}^{Rp}_{i_r}$ are only labeled by clauses (Rap);

- $\mathcal{D}'_1, \ldots, \mathcal{D}'_q$ are derivations of facts $\mathrm{att}_i(U'_1), \ldots, \mathrm{att}_i(U'_q)$ respectively at step $\tau_1, \ldots, \tau_q$ with for all $k \in \{1, \ldots, q\}$, $\tau_k \leq \tau$, root of U'_k is not in $\mathcal{F}_{data}$ and $\mathrm{att}_i \in \mathcal{S}_p$ implies $\tau_k < \tau$.

Notice that if the incoming edge of η is labeled by $\mathrm{att}_i(f(M_1, \ldots, M_m))$ with $f \in \mathcal{F}_{data}$ then either η is an unlabeled leaf of the partial derivation $\mathcal{D}_p$ or η is labeled by a clause $R_\eta = (H_\eta \to C_\eta)$. In the latter case, we know that there exists σ' such that $C_\eta\sigma' = \mathrm{att}_i(f(M_1, \ldots, M_m))$. However, since the clauses of $\mathbb{C}$ are simplified, the rule DataCl cannot be applied on R_η. Thus, either R_η is the clause $\mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_m) \to \mathrm{att}_i(f(x_1, \ldots, x_m))$ or $\mathcal{C}_\eta = \mathrm{att}_i(x)$ for some variable x. In the latter case, since R_η is well originated and $\mathcal{C}_\eta \in \mathbb{F}_{usel}$, we deduce that there exists j such that $\mathrm{att}_j(x) \in H_\eta$. By Definition 20 of a partial derivation, we deduce that R_η is in fact the clause $\mathrm{att}_{i-1}(x) \to \mathrm{att}_i(x)$.

We can now prove our statement by induction as mentioned. In the base case, η is a leaf of the partial derivation $\mathcal{D}_p$. We do a small case analysis: (i) The incoming edge of η is labeled by F_η, τ_η such that $F_\eta \neq \mathrm{att}_i(f(M_1, \ldots, M_m))$ for all $f \in \mathcal{F}_{data}$ and terms $M_1, \ldots, M_m$: In such a case, either η is labeled by a clause without hypothesis in $\mathcal{D}_p$, hence by definition of the derivation context $\mathcal{D}'$, $\tau_\eta = \tau$ and $\mathit{pred}(F_\eta) \notin \mathcal{S}_p$; or the corresponding node of η in $\mathcal{D}_p$ is in fact an unlabeled leaf. In the latter case, there exists $k \in \{1, \ldots, n\}$ such that $\mathcal{D}_\eta = \mathcal{D}_k$ and $\tau_\eta = \tau_k$. Since by hypothesis $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_\eta$, $\tau_\eta \leq \tau$ and if $\mathit{pred}(F_\eta) \in \mathcal{S}_p$ then $\tau_\eta < \tau$. Thus, the result holds by taking the empty context for $\mathcal{D}^{\mathcal{F}_{data}}$, $q = 1$, $r = 0$ and $\mathcal{D}'_1 = \mathcal{D}_\eta$. (ii) The incoming edge of η is labeled by F_η, τ_η such that $F_\eta = \mathrm{att}_i(f(M_1, \ldots, M_m))$ for some $f \in \mathcal{F}_{data}$ and terms $M_1, \ldots, M_m$: By the remark in the previous paragraph, we deduce that in such a case, η is necessarily unlabeled in $\mathcal{D}_p$ meaning that $\mathrm{att}_i(f(M_1, \ldots, M_m)) \in \mathbb{F}_{us}(\mathcal{D}_p)$ and so there exists $k \in \{1, \ldots, n\}$ such that $\mathcal{D}_\eta = \mathcal{D}_k$. Thus the result holds by taking $\mathcal{D}^{\mathcal{F}_{data}}$ the empty context, $q = 0$, $r = 1$, $i_1 = k$ and $\mathcal{D}^{Rp}_{i_1}$ the empty context.

In the inductive step, once again if the incoming edge is not labeled by $\mathrm{att}_i(f(M_1, \ldots, M_m))$ for some $f \in \mathcal{F}_{data}$, $M_1, \ldots, M_m$ then we obtain the result by taking the empty context for $\mathcal{D}^{\mathcal{F}_{data}}$, $q = 1$, $r = 0$ and $\mathcal{D}'_1 = \mathcal{D}_\eta$. Otherwise one of the following two cases occurs:

- η is labeled by $\mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_m) \to \mathrm{att}_i(f(x_1, \ldots, x_m))$: In such a case, we have $\mathcal{D}^1_\eta$, …, $\mathcal{D}^m_\eta$ derivations respectively of $\mathrm{att}_i(M_1)$, …, $\mathrm{att}_i(M_m)$ on which we can apply our inductive hypothesis to directly obtain our result since the clause $\mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_m) \to \mathrm{att}_i(f(x_1, \ldots, x_m))$ is the clause (Rf_f).

- η is labeled by $\mathrm{att}_{i-1}(x) \to \mathrm{att}_i(x)$: By Lemma 30, we deduce that $\mathcal{D}_p$ contains a branch from the node corresponding to η that is a sequential application of rules $\mathrm{att}_j(x) \to \mathrm{att}_{j+1}(x)$ and there exists $i' \leq i$ such that $\mathrm{att}_{i'}(x) \in \mathbb{F}_{usel}(\mathcal{D}_p)$. Thus there exists $k \in \{1, \ldots, n\}$ such that the incoming edge of $\mathcal{D}_k$ is labeled by the fact $\mathrm{att}_{i'}(f(M_1, \ldots, M_m))$. Hence the result holds by taking $\mathcal{D}^{\mathcal{F}_{data}}$ the empty context, $q = 0$, $r = 1$ and $i = i_1$ and $\mathcal{D}^{Rp}_1$ the context corresponding to the sequential application of rules $\mathrm{att}_j(x) \to \mathrm{att}_{j+1}(x)$.

We showed that $\mathcal{D}_\eta = \mathcal{D}^{\mathcal{F}_{data}}[\mathcal{D}'_1, \ldots, \mathcal{D}'_p, \mathcal{D}^{Rp}_{i_1}[\mathcal{D}_{i_1}], \ldots, D^{Rp}_{i_r}[\mathcal{D}_{i_r}]]$. However, for all $k \in \{1, \ldots, r\}$, the derivation $\mathcal{D}_{i_k}$ can be a derivation of some fact $\mathrm{att}_{i'}(U)$ with U being rooted by a data constructor function symbol. However, coming back to the hypotheses of the lemma, we know that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{i_k}$. Hence, if U is rooted by $g \in \mathcal{F}_{data}$, we deduce that the root of $\mathcal{D}_{i_k}$ is labeled by the clause (Rf_g), i.e. $\mathrm{att}_{i'}(x_1) \wedge \ldots \wedge \mathrm{att}_{i'}(x_m) \to \mathrm{att}_{i'}(g(x_1, \ldots, x_m))$ (Item 4 of Definition 18). Thus by applying a small induction on the size of $\mathcal{D}_{i_k}$, we can show that $\mathcal{D}_{i_k} = \mathcal{D}^{\mathcal{F}_{data}}_{i_k}[\mathcal{D}^{i_k}_1, \ldots, \mathcal{D}^{i_k}_{\ell_{i_k}}]$ where for all $\ell \in \{1, \ldots, \ell_{i_k}\}$,

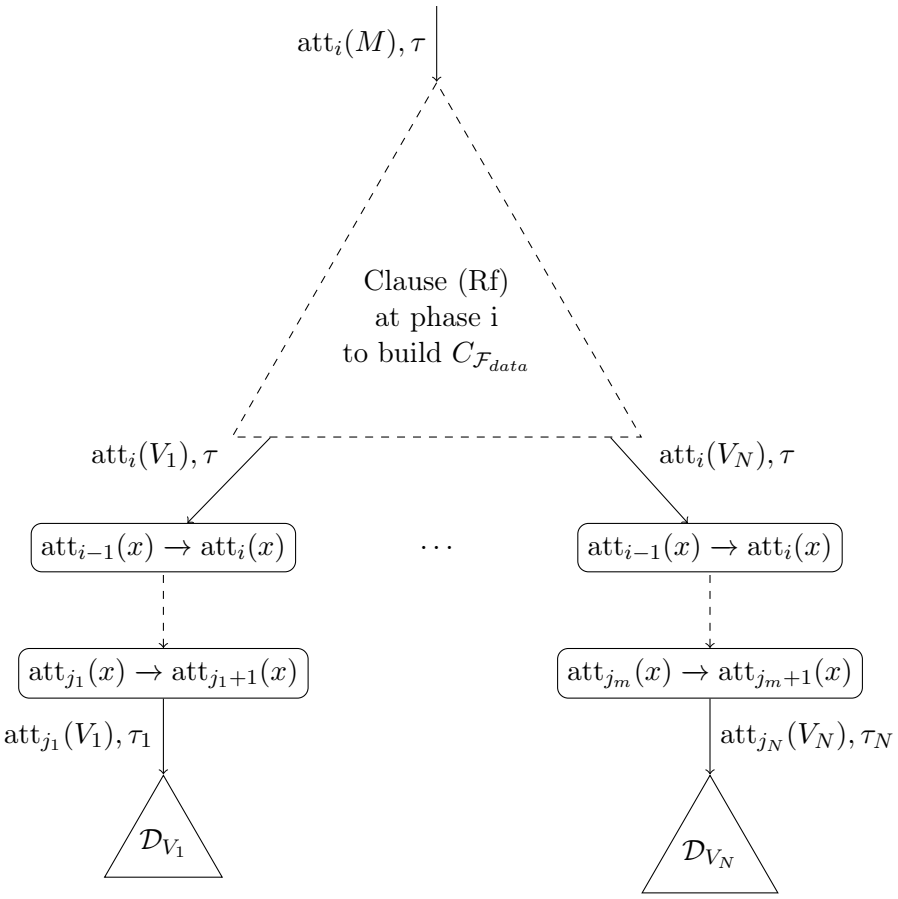
- $\mathcal{D}_\ell^{i_r}$ is a derivation of $\mathrm{att}_{i'}(U_\ell^{i_k})$ at step $\tau_\ell^{i_k}$ where $U_\ell^{i_k}$ is not rooted by a data constructor function symbol;
- $\tau_\ell^{i_k} \leq \tau$ and if $\mathrm{att}_{i'} \in \mathcal{S}_p$ then $\tau_\ell^{i_k} < \tau$;
- nodes of $\mathcal{D}_{i_k}^{\mathcal{F}_{data}}$ are only labeled by clauses (Rf_g) for some $g \in \mathcal{F}_{data}$.

Thus, by gathering $U'_1, \ldots, U'_q, U_1^{i_1}, \ldots, U_{\ell_{i_1}}^{i_1}, \ldots, U_1^{i_r}, \ldots, U_{\ell_{i_r}}^{i_r}$ and renaming them $V_1, \ldots, V_N$, we deduce that $\mathcal{D}_\eta$ can be written as $\mathcal{D}'_c[\mathcal{D}_{V_1}, \ldots, \mathcal{D}_{V_N}]$ where

- the nodes of $\mathcal{D}'_c$ are either labeled by a clause (Rap) or (Rf_g) for some $g \in \mathcal{F}_{data}$;
- for all $k \in \{1, \ldots, N\}$, the incoming edge of $\mathcal{D}_{V_k}$ is labeled by $\mathrm{att}_{j_k}(V_k), \tau_k$ where the root of V_k is not a data constructor symbol and either $\tau_k \leq \tau$ and $\mathrm{att}_{j_k} \notin \mathcal{S}_p$; or $\tau_k < \tau$, $T, \tau_k \vdash_{IO}^{n_{IO}} \mathrm{att}_{j_k}(V_k)$, $\mathrm{att}_{j_k} \in \mathcal{S}_p$ and for all $j_k < j \leq i$, $\mathrm{att}_j \notin \mathcal{S}_p$;
- $M = C_{\mathcal{F}_{data}}[V_1, \ldots, V_N]$ where $C_{\mathcal{F}_{data}}$ is a term context composed only of data constructor function symbols (recall the $\mathrm{att}_i(M)$ was the fact label of the incoming edge of the node η).

We transform the derivation $\mathcal{D}'_c[\mathcal{D}_{V_1}, \ldots, \mathcal{D}_{V_N}]$ into a derivation $\mathcal{D}'_\eta$ by first applying the clauses (Rap) to derive the fact $\mathrm{att}_i(V_k)$ at step τ and second by applying the clauses $\mathrm{att}_i(x_1) \wedge \ldots \wedge \mathrm{att}_i(x_\ell) \to \mathrm{att}_i(g(x_1, \ldots, x_\ell))$ with g occurring in $C_{\mathcal{F}_{data}}$ to derive the fact $\mathrm{att}_i(M)$ at step τ.

The derivation $\mathcal{D}'_\eta$



Note that the derivation $\mathcal{D}'_\eta$ satisfies all items of Definition 18 and in particular Item 4 since the phase clauses (Rap) are only applied on attacker facts whose term is not rooted by a data constructor function symbol. Therefore, to build $\mathcal{D}$, we replace in $\mathcal{D}'$ the derivation $\mathcal{D}'_c[\mathcal{D}_{V_1}, \ldots, \mathcal{D}_{V_N}]$ by the derivation $\mathcal{D}'_\eta$, which allows us to obtain that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$.

Finally, since all clauses in $\mathcal{D}_p$ are selection free then so are the clauses in $\mathcal{D}_c[\__1, \ldots, \__n]$. Moreover, in the second part of proof, we only added clauses (Rf) and (Rap) which are also selection free. Thus, if $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are selection free then $\mathcal{D}_{V_1}, \ldots, \mathcal{D}_{V_N}$ are selection free which implies that $\mathcal{D}$ is selection free. Thus we conclude. □

Lemma 35. *Let $\mathbb{C}$ be a set of well originated, simplified clauses containing $\mathbb{C}_{std}$. Let $R \notin \mathbb{C}_{std}$ be a well originated, simplified clause. Let $\mathcal{D}_{F_0} = \mathcal{D}[\mathcal{D}_R]$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \{R = H \wedge \phi \to F\}$ such that $\{\!\{\tau_0\}\!\} \leq_m \mathcal{M}$ and all nodes in $\mathcal{D}_R$ are labeled by clauses from $\mathbb{C}$ except the root that is labeled by R and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$.*

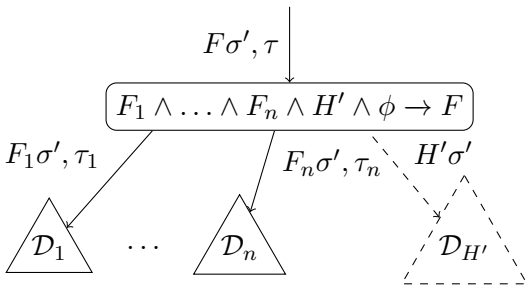
If there exists $\mathcal{D}_p$ a partial derivation of F from $\mathbb{C}' = \{R' \in \mathbb{C} \mid \mathrm{sel}(R') = \emptyset\}$ such that $\mathbb{F}_{us}(\mathcal{D}_p) \to F \sqsupseteq R$ and $pred(\mathbb{F}_s(\mathcal{D}_p)) \cap \mathcal{S}_p = \emptyset$ then there exists a derivation $\mathcal{D}'_R$ from $\mathbb{C}$ such that:

1. *$T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}'_R]$*
2. *if $\mathcal{D}_R$ is selection free then $\mathcal{D}'_R$ is selection free.*

Proof. Let us consider some notation. Let us write $\mathbb{F}_{us}(\mathcal{D}_p) = U_1 \wedge \ldots U_n \wedge \phi_u$ where $U_1, \ldots, U_n$ are facts and ϕ_u is a formula. Moreover, since $\mathbb{F}_{us}(\mathcal{D}_p) \to F \sqsupseteq H \wedge \phi \to F$, there exist a substitution σ and $F_1, \ldots, F_n, H'$ such that $F\sigma = F$, $H = F_1 \wedge \ldots \wedge F_n \wedge H' \wedge \phi$, $U_i\sigma = F_i$ for all i and $\phi \models \phi_u\sigma$.

Since $\mathcal{D}_R$ is rooted by R, there exist a conjunction of derivations $\mathcal{D}_{H'}$, some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and a substitution σ' such that $\mathcal{D}_R$ is a derivation of $F\sigma'$ at some step τ, $\vdash^{n_{IO}}_{IO} \phi\sigma'$ and $\mathcal{D}_{H_2}$ is a conjunction derivation of $H'\sigma'$ and $\mathcal{D}_i$ is a derivation of $F_i\sigma'$ at some step τ_i for all $i \in \{1, \ldots, n\}$. Since $R \notin \mathbb{C}_{std}$ and in particular, R is not (Rl) nor a projection clause of a data constructor function symbol, we deduce from $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}_R]$ that for all $i \in \{1, \ldots, n\}$, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$, $\tau_i \leq \tau$ and if $pred(U_i) \in \mathcal{S}_p$ then $\tau_i < \tau$. Note that $\vdash^{n_{IO}}_{IO} \phi\sigma'$ and $\phi \models \phi_u\sigma$ implies $\vdash^{n_{IO}}_{IO} \phi_u\sigma\sigma'$.

The derivation $\mathcal{D}_R$



We can apply Lemma 34 with the substitution $\sigma\sigma'$, the steps $\tau_1, \ldots, \tau_n$ and the derivations $\mathcal{D}_1, \ldots, \mathcal{D}_n$ to obtain that there exists a derivation $\mathcal{D}'_R$ of $F\sigma\sigma' = F\sigma'$ at step τ from $\mathbb{C}$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_R$ and if $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are selection free then $\mathcal{D}'_R$ is selection free. Hence, we directly obtain that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}[\mathcal{D}'_R]$. Moreover, if $\mathcal{D}_R$ is selection free then $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are selection free thus $\mathcal{D}'_R$ is selection free. □

Corollary 5. *Let $\mathbb{C}$ be a set of well originated, simplified clauses such that $\mathbb{C}_{std} \subseteq \mathbb{C}$. Let $\mathbb{C}'$ the resulting set by application of the rule GRed($\mathcal{S}_p$) to $\mathbb{C}$. Let $\mathcal{D}_{F_0}$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C}$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_F$. There exists a derivation $\mathcal{D}'_{F_0}$ of F_0 at step τ_0 from $\mathbb{C}'$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_{F_0}$.*

Proof. The proof is done by induction on the number of nodes labeled by the clause $H \to F$ on which GRed($\mathcal{S}_p$) is applied, the inductive step being a direct application of Lemma 35. □

Corollary 6. *Let $\mathbb{C}$ be a set of well originated, simplified clauses such that $\mathbb{C}_{std} \subseteq \mathbb{C}$. Let $\mathcal{D}_{F_0}$ be a derivation of some fact F_0 at step τ_0 from $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{F_0}$. There exists a derivation $\mathcal{D}'_{F_0}$ of F_0 at step τ_0 from $\mathsf{condense}_{\mathcal{S}_p}(\mathbb{C}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_{F_0}$.*

Proof. Direct application of Corollaries 4 and 5. □

C.4 Main proof

Theorem 2. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. Let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant restrictions. Let $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \to_i)_{|\mathcal{R}}$.*

For all well originated sets of clauses $\mathbb{C}$ containing $\mathbb{C}_{std}$, for all derivations $\mathcal{D}$ of F at step τ from $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, (\tau))$ and $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$, there exists a derivation $\mathcal{D}'$ of F at step τ from $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\mathbb{C}) \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}'$.

Proof. The first part of the proof is almost a direct consequence of Corollaries 2, 3 and 6.

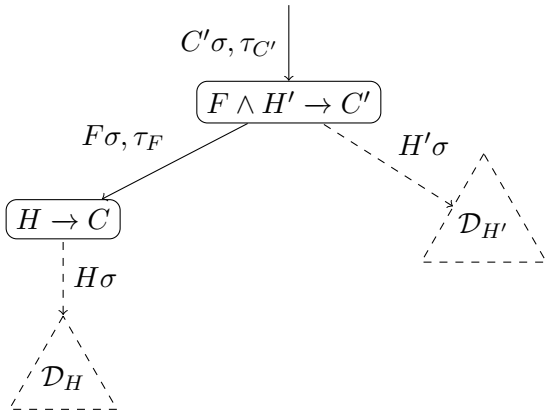
The first step of $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ consists of generating $\mathbb{C}_1 = \mathsf{condense}_{\mathcal{S}_p}(\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}))$. Since $\mathbb{C}$ is well originated and $\mathbb{C}_{std} \subseteq \mathbb{C}$, we can apply Lemma 31 and corollaries 3 and 6 to obtain that there exists a derivation $\mathcal{D}^1_F$ of F at step τ from $\mathbb{C}_1 \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}^1_F$. The second step of the saturation generates a new clause R by application of the resolution rule Res. Hence $\mathcal{D}^1_F$ is a derivation from $\mathbb{C}_1 \cup \{R\} \cup \mathbb{C}_e(T)$. The third step of the saturation computes the set $\mathbb{C}_2 = \mathsf{condense}_{\mathcal{S}_p}(\mathbb{C}_1 \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}))$. Thus by applying Corollaries 2 and 6, we obtain that there exists a derivation $\mathcal{D}^2_F$ of F at step τ from $\mathbb{C}_2 \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}^2_F$. Since the second and third steps are repeated until a fix point is reached, denoted $\mathbb{C}_{sat}$, we can deduce that there exists a derivation $\mathcal{D}'_F$ of F at step τ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}'_F$.

By definition, $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}) = \{R \in \mathbb{C}_{sat} \mid \mathrm{sel}(R) = \emptyset\}$. Hence, we still have to show that we can build a selection free derivation $\mathcal{D}'$ of F at step τ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$. To do so, given a derivation $\mathcal{D}$, we denote by $|\mathcal{D}|_{\mathrm{sel}}$ the number of nodes in $\mathcal{D}$ labeled by a clause R such that $\mathrm{sel}(R) \neq \emptyset$.

Let us consider the derivation $\mathcal{D}'$ of F at step τ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ that is minimal for $(|\mathcal{D}'|_{\mathrm{sel}}, |\mathcal{D}'|)$ (using the lexicographic order). We show by contradiction that $\mathcal{D}'$ is selection free, i.e. $|\mathcal{D}'|_{\mathrm{sel}} = 0$.

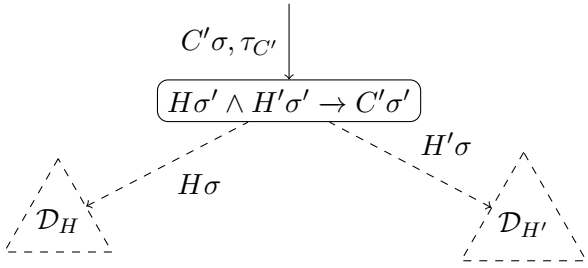
Assume that $|\mathcal{D}'|_{\mathrm{sel}} \neq 0$. Hence $\mathcal{D}' = \mathcal{D}_c[\mathcal{D}_R]$ for some derivation context $\mathcal{D}_c$ and some derivation $\mathcal{D}_R$ such that $\mathcal{D}_R$ is labeled by some clause $R = (F \wedge H' \to C')$ with $F \in \mathrm{sel}(R)$. By denoting $\mathbb{C}_{sat} = \mathbb{C}'_{sat} \cup \{R\}$, we can assume by minimality on the size of $\mathcal{D}_R$ that all nodes of $\mathcal{D}_R$ except the root are labeled by clauses of $\mathbb{C}'_{sat} \cup \mathbb{C}_e(T)$ with no selected hypothesis. By definition of a derivation, we deduce that $\mathcal{D}_R$ is of the following form:

The derivation $\mathcal{D}_R$



where σ is a substitution, $\mathcal{D}_H$ and $\mathcal{D}_{H'}$ are respectively conjunction derivations of $H\sigma$ and $H'\sigma$. Note that since F is selectable, F cannot be an event. Thus $(H \to C) \notin \mathbb{C}_e(T)$ and so $H \to C \in \mathbb{C}_{sat}$. Moreover since $F\sigma$ is the label of the incoming edge of the node labeled by $H \to C$, we deduce that $C\sigma = F\sigma$. Thus, we can compute σ' the most general unifier of C and F and apply the rule Res to obtain the clause $R' = H\sigma' \wedge H'\sigma' \to C'\sigma'$, since $\text{sel}(H \to C) = \emptyset$ because all nodes of $\mathcal{D}_R$ except the root are labeled by clauses with no selected hypothesis. Thus we can build the following derivation $\mathcal{D}'_R$ of $C'\sigma$ at step τ'_C:

The derivation $\mathcal{D}_R$



Notice that $|\mathcal{D}'_R| < |\mathcal{D}_R|$ and so $|\mathcal{D}_c[\mathcal{D}'_R]| < |\mathcal{D}'|$. Moreover, $|\mathcal{D}_c[\mathcal{D}'_R]|_{\text{sel}} \leq |\mathcal{D}'|_{\text{sel}}$ (the clause $H\sigma' \wedge H'\sigma' \to C'\sigma'$ may not be selection free) and so $(|\mathcal{D}_c[\mathcal{D}'_R]|_{\text{sel}}, |\mathcal{D}_c[\mathcal{D}'_R]|) < (|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$.

Finally, since $\text{sel}(H \to C) = \emptyset$, the clause $H \to C$ cannot be a clause (Rf) for a projection of a data constructor function symbol. On the other hand, since $F \in \text{sel}(F \wedge H' \to C')$, the clause $F \wedge H' \to C'$ is not a clause (Rf) for a data constructor function symbol. Therefore, we deduce from $T, \mathcal{S}_p, n \vdash \mathcal{D}'$ that $T, \mathcal{S}_p, n \vdash \mathcal{D}_c[\mathcal{D}'_R]$.

Notice that $\mathcal{D}'_R$ is strictly selection free. Thus by applying Lemma 32, we deduce that there exist two derivation contexts $\mathcal{D}_\pi[\_], \mathcal{D}'_c[\_]$ and a derivation $\mathcal{D}''_R$ such that:

- $\mathcal{D}_c = \mathcal{D}'_c[\mathcal{D}_\pi[\_]]$
- $\mathcal{D}_\pi[\_]$ is a context derivation with a unique hole and whose nodes are labeled by a clause $\text{att}_n(f(x_1, \ldots, x_m)) \to \text{att}_n(x_i)$ for some $f \in \mathcal{F}_{data}$ and $i \in \{1, \ldots, m\}$
- $\mathcal{D}''_R$ is a derivation from $\mathbb{C}'_{sat} \cup \mathbb{C}_e(T) \cup \mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R'\})$ such that all nodes except the root are labeled by clauses from $\mathbb{C}'_{sat} \cup \mathbb{C}_e(T)$.

- $T, \mathcal{S}_p, n \vdash \mathcal{D}'_c[\mathcal{D}''_R]$
- $|\mathcal{D}''_R| \leq |\mathcal{D}_\pi[\mathcal{D}'_R]|$
- $\mathcal{D}''_R$ is strictly selection free.

Thus, we deduce that $|\mathcal{D}'_c[\mathcal{D}''_R]| \leq |\mathcal{D}'_c[\mathcal{D}_\pi[\mathcal{D}'_R]]| = |\mathcal{D}_c[\mathcal{D}'_R]| < |\mathcal{D}'|$ and $|\mathcal{D}'_c[\mathcal{D}''_R]|_{\text{sel}} \leq |\mathcal{D}'_c|_{\text{sel}} + 1 \leq |\mathcal{D}_c|_{\text{sel}} + 1 \leq |\mathcal{D}'|_{\text{sel}}$ and so $(|\mathcal{D}'_c[\mathcal{D}''_R]|_{\text{sel}}, |\mathcal{D}'_c[\mathcal{D}''_R]|) < (|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$

Since $\mathbb{C}_{sat}$ was obtained by applying the saturation steps until a fix point is reached, we deduce that the clause R'' labeling the root of $\mathcal{D}''_R$ satisfies one of the following properties:

- R'' is in $\mathbb{C}_{sat}$: Such a case is impossible as $(|\mathcal{D}'_c[\mathcal{D}''_R]|_{\text{sel}}, |\mathcal{D}'_c[\mathcal{D}''_R]|) < (|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$ would contradict the minimality of $|\mathcal{D}'|$ w.r.t. $(|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$.
- there exists a clause R_s in $\mathbb{C}_{sat}$ that subsumes R'': By Lemma 33, there exists a derivation $\mathcal{D}_s$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}'_c[\mathcal{D}_s]$, $|\mathcal{D}_s| \leq |\mathcal{D}''_R|$ and $\mathcal{D}_s$ is strictly selection free. Thus we have once again $|\mathcal{D}'_c[\mathcal{D}_s]| < |\mathcal{D}'|$ and $|\mathcal{D}'_c[\mathcal{D}_s]|_{\text{sel}} \leq |\mathcal{D}'|_{\text{sel}}$ and so $(|\mathcal{D}'_c[\mathcal{D}_s]|_{\text{sel}}, |\mathcal{D}'_c[\mathcal{D}_s]|) < (|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$ which contradicts the minimality of $|\mathcal{D}'|$ w.r.t. $(|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$.
- $\text{sel}(R'') = \emptyset$ and was removed by application of the rule $\text{GRed}(\mathcal{S}_p)$. But in such a case, $\mathcal{D}''_R$ is selection free. Furthermore, by Lemma 35, there exists a derivation $\mathcal{D}_r$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}'_c[\mathcal{D}_r]$ and $\mathcal{D}_r$ is selection free. Thus, we deduce that $|\mathcal{D}'_c[\mathcal{D}_r]|_{\text{sel}} < |\mathcal{D}'|_{\text{sel}}$ which entails $(|\mathcal{D}'_c[\mathcal{D}_r]|_{\text{sel}}, |\mathcal{D}'_c[\mathcal{D}_r]|) < (|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$. This once again contradicts the minimality of $|\mathcal{D}'|$ w.r.t. $(|\mathcal{D}'|_{\text{sel}}, |\mathcal{D}'|)$.

Since all possible cases end with a contradiction, we can conclude that $|\mathcal{D}'|_{select} = 0$ and so $\mathcal{D}'$ is a derivation of F at step τ from $\mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}) \cup \mathbb{C}_e(T)$. □

D Proof of Theorem 4

The proof of Theorem 4 will follow closely the proof of Theorem 2 since both focus on the saturation procedure, the former being on ordered clauses. Since some simplification rules cannot be applied when computing $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}, \mathbb{C}_{sat})$ (e.g. (Taut), (DataCl)), the lemmas will be simpler to state and to prove even though they will contain additional arguments to handle the ordering functions.

As for the proof of Theorem 2, we consider a preamble to all lemmas in this section:

Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates containing the event predicate. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of ProVerif *IO-n_{IO}-compliant lemmas. Let $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$ and $\mathcal{M}$ be a multiset such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \mathcal{M})$ holds. Let $\mathbb{C}_{sat}$ be a set of well originated, simplified, selection free clauses containing the selection free clauses of $\mathbb{C}_{std}$.*

Once again, this preamble corresponds to the hypotheses of Theorem 4.

D.1 Soundness of $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$

Lemma 36. *Let R be an ordered clause satisfying $\mathcal{S}_p$. Let $\mathcal{D}$ be an ordered derivation of $\bigwedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $\{\!\{\tau_1, \ldots, \tau_m\}\!\} \leq_m \mathcal{M}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. All clauses in $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ satisfy $\mathcal{S}_p$ and there exists an ordered derivation $\mathcal{D}'$ of $\bigwedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and $|\mathcal{D}'| \leq |\mathcal{D}|$.*

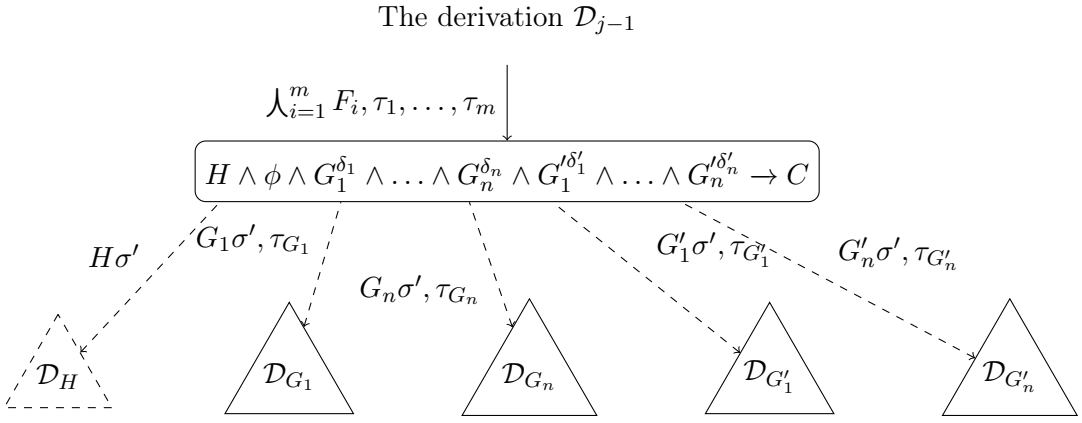
Proof. By definition, $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ is the repeated application on $\{R\}$ of the rules Red_o, Att_o, $\text{R}_{\phi,o}$, DataHyp_o, Nat_o, $\text{Lem}_o(\mathcal{L}, \mathcal{S}_p)$ and $\text{Ind}_o(\mathcal{L}_i, \mathcal{S}_p)$. Therefore, the computation of $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$ can be seen as a sequence $\mathbb{C}_0, \ldots, \mathbb{C}_N$ where $\mathbb{C}_N = \mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\})$, $\mathbb{C}_0 = \{R\}$ and for all $j \in \{1, \ldots, N\}$, $\mathbb{C}_j$ is obtained from $\mathbb{C}_{j-1}$ by application of one the rules Red_o, Att_o, $\text{R}_{\phi,o}$, DataHyp_o, Nat_o, $\text{Lem}_o(\mathcal{L}, \mathcal{S}_p)$ and $\text{Ind}_o(\mathcal{L}_i, \mathcal{S}_p)$.

The proof of the lemma is thus done by induction on N by showing that for all $j \in \{0, \ldots, N\}$, we can build an ordered derivation $\mathcal{D}_j$ of $\bigwedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_j$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$.

Base case $j = 0$: Since $\mathbb{C}_0 = \{R\}$, the result directly holds by taking $\mathcal{D}_0 = \mathcal{D}$.

Inductive step $j > 0$: In such a case, we apply our inductive hypothesis on $j - 1$ which gives us the existence of the ordered derivation $\mathcal{D}_{j-1}$ of $\bigwedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_j$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$. Let us look at the transformation applied on $\mathbb{C}_{j-1}$ to obtain $\mathbb{C}_j$. Notice that all the transformation rules only affect one clause at a time. Thus, there exists R' such that $\mathbb{C}_{j-1} = \mathbb{C}'_{j-1} \cup \{R'\}$ and $\mathbb{C}_j = \mathbb{C}'_{j-1} \cup \mathbb{C}'$ where $\mathbb{C}'$ is the result of the application of a transformation rule on R'. Hence if the child of the root of the derivation $\mathcal{D}_{j-1}$ is labeled by a clause of $\mathbb{C}'_{j-1}$ then the result trivially holds by taking $\mathcal{D}_j = \mathcal{D}_{j-1}$. Otherwise, the child of the root of the derivation $\mathcal{D}_{j-1}$ is labeled by R'. We do a case analysis on the transformation rule applied on R':

- Rule Red_o: In such a case, R' is of the form $H \wedge \phi \wedge \bigwedge_{k=1}^n G_k^{\delta_k} \wedge G_k'^{\delta'_k} \rightarrow C$ such that $G'_k\sigma = G_k$ and either $G'_k = G_k$ or $\delta'_k \sqsupseteq_o \delta_k$ for all $k \in \{1, \ldots, n\}$, $\phi \models \phi\sigma$ and $\text{dom}(\sigma) \cap fv(H, C, G_1, \ldots, G_n) = \emptyset$. Moreover, by Definition 26, we know that there exist a substitution σ, some derivations $\mathcal{D}_H, \mathcal{D}_{G_1}, \ldots, \mathcal{D}_{G_n}, \ldots, \mathcal{D}_{G'_1}, \ldots, \mathcal{D}_{G'_n}$ and some steps $\tau_{G_1}, \ldots, \tau_{G_n}, \tau_{G'_1}, \ldots, \tau_{G'_n}$ such that $C\sigma' = \bigwedge^m_{i=1} F_i$ and the derivation $\mathcal{D}_{j-1}$ has the following form:



Moreover, we also know that for all $i \in \{1, \ldots, m\}$, for all $k \in \{1, \ldots, n\}$, if $\delta_k(i)$ is defined then $\tau_{G_k}\ \delta_k(i)\ \tau_i$. Similarly, if $\delta'_k(i)$ is defined then $\tau_{G'_k}\ \delta'_k(i)\ \tau_i$.

For all $k \in \{1, \ldots, n\}$, we define a new derivation $\mathcal{D}'_k$ and a step τ'_k defined as follows: if $G'_k = G_k$ and $\tau_{G'_k} < \tau_{G_k}$ then $\mathcal{D}'_k = \mathcal{D}_{G'_k}$ and $\tau'_k = \tau_{G'_k}$ else $\mathcal{D}'_k = \mathcal{D}_{G_k}$ and $\tau'_k = \tau_{G_k}$.

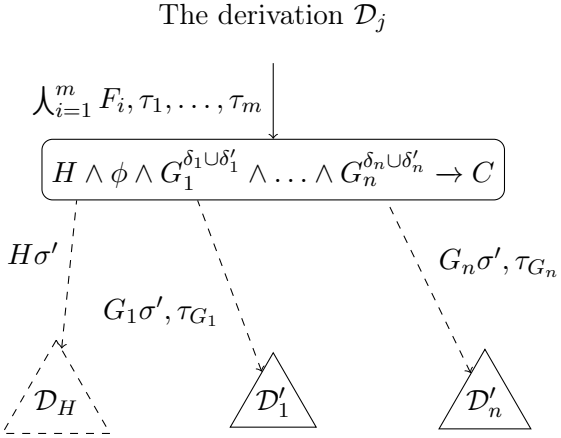
First notice that $\mathcal{D}'_k$ is a derivation of $G_k\sigma'$ at step τ'_k. Indeed, when $G'_k = G_k$ and $\tau_{G'_k} < \tau_{G_k}$, we trivially have $G'_k\sigma' = G_k\sigma'$ and D'_k is a derivation of $G'_k\sigma'$. Otherwise $\mathcal{D}'_k = \mathcal{D}_{G_k}$ and $\tau'_k = \tau_{G_k}$ hence the result directly holds.

Second, if $pred(G_k) \notin \mathcal{S}_p$ then $pred(G'_k) \notin \mathcal{S}_p$ since $G'_k\sigma = G_k$. Thus, with R' satisfying $\mathcal{S}_p$, we deduce that if $pred(G_k) \notin \mathcal{S}_p$ then for all $i \in \{1, \ldots, m\}$, $\delta_k(i) \neq <$ and $\delta'_k(i) \neq <$. By definition of $(\delta_k \cup \delta'_k)(i)$, we therefore deduce that if $pred(G_k) \notin \mathcal{S}_p$ then $(\delta_k \cup \delta'_k)(i) \neq <$. Hence, the rule $H \wedge \phi \wedge G_1^{\delta_1 \cup \delta'_1} \wedge \ldots \wedge G_n^{\delta_n \cup \delta'_n} \rightarrow C$ satisfies $\mathcal{S}_p$.

Let us show that for all $i \in \{1, \ldots, m\}$, if $(\delta_k \cup \delta'_k)(i)$ is defined then $\tau'_k\ (\delta_k \cup \delta'_k)(i)\ \tau_i$. By definition of $\delta_k \cup \delta'_k$, if $(\delta_k \cup \delta'_k)(i)$ is defined then $\delta_k(i)$ or $\delta'_k(i)$ are defined. Moreover, $(\delta_k \cup \delta'_k)(i) = <$ implies $< \in \{\delta_k(i), \delta'_k(i)\}$. We do a case analysis:

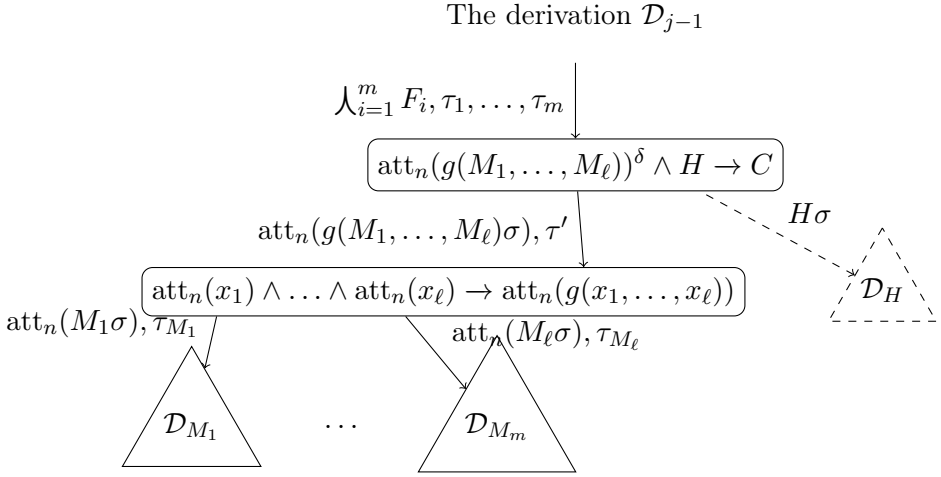
1. if $\delta_k(i)$ is defined and $\delta_k(i) = <$. In such case, we know that $\tau_{G_k} < \tau_i$. If $\tau'_k = \tau_{G_k}$ then the result directly holds, else $\tau'_k = \tau_{G'_k}$ and $\tau_{G'_k} < \tau_{G_k}$ by definition, implying $\tau'_k < \tau_i$. Hence the result holds.
2. if $\delta'_k(i)$ is defined and $\delta'_k(i) = <$. In such a case, we know that $\tau_{G'_k} < \tau_i$. If $\tau'_k = \tau_{G'_k}$ then the result directly holds. Otherwise, either $G'_k = G_k$ and $\tau_{G_k} \leq \tau_{G'_k}$; or $\delta'_k \sqsupseteq_o \delta_k$. In the former case, $\tau_{G'_k} < \tau_i$ and $\tau_{G_k} \leq \tau_{G'_k}$ implies $\tau_{G_k} < \tau_i$ and so $\tau'_k < \tau_i$. In the latter case, by definition of the subsumption, $\delta'_k(i)$ being defined implies $\delta_k(i)$ being defined. Moreover, since $\delta'_k(i) = <$ then $\delta_k(i) = <$. Hence we deduce that $\tau_{G_k} < \tau_i$ and so $\tau'_k < \tau_i$.
3. if $\delta_k(i) = \leq$ and $\delta'_k(i) \neq <$: In such case, we know that $\tau_{G_k} \leq \tau_i$. The rest of the proof is the same as in Case 1.
4. if $\delta'_k(i) = \leq$ and $\delta_k(i) \neq <$: In such a case, we know that $\tau_{G'_k} \leq \tau_i$. . If $\tau'_k = \tau_{G'_k}$ then the result directly holds. Otherwise, either $G'_k = G_k$ and $\tau_{G_k} \leq \tau_{G'_k}$; or $\delta'_k \sqsupseteq_o \delta_k$. In the former case, $\tau_{G'_k} \leq \tau_i$ and $\tau_{G_k} \leq \tau_{G'_k}$ implies $\tau_{G_k} \leq \tau_i$ and so $\tau'_k \leq \tau_i$. In the latter case, by definition of the subsumption, $\delta'_k(i)$ being defined implies $\delta_k(i)$ being defined. Moreover, since $\delta'_k(i) = \leq$ then $\delta_k(i) = \leq$ or $\delta_k(i) = <$. In both cases, we deduce that $\tau_{G_k} \leq \tau_i$ and so $\tau'_k \leq \tau_i$.

We can therefore build the derivation $\mathcal{D}_j$ as follows:



- Rule Att_o: In such a case, R' is of the form $\text{att}_i(x)^\delta \wedge H \wedge \phi \rightarrow C$ where x does not appear in H and C. Since we only remove the $\text{att}_i(x)^\delta$ from the clause without changing the rest of the clause, the result directly holds.

- Rule DataHyp_o: In such a case, R' is of the form $\text{att}_n(g(M_1, \ldots, M_\ell))^\delta \wedge H \rightarrow C$ where $g \in \mathcal{F}_{data}$. Moreover, we transform the clause R' into the clause $R'' = \text{att}_n(M_1)^{\delta'} \wedge \ldots \wedge \text{att}_n(M_\ell)^{\delta'} \wedge H \rightarrow C$ where $\delta' = \delta^<$ if $\text{att}_n \in \mathcal{S}_p$ else $\delta' = \delta$. By Definition 26, we know that the child η of the root is labeled by R' and possesses a child η' such that $\eta \xrightarrow{\text{att}_n(g(M_1,\ldots,M_\ell)\sigma),\tau'} \eta'$ for some σ, τ'. Moreover, Definition 26 also indicates that the sub-derivation $\mathcal{D}'$ rooted in η' satisfies $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. Hence, since $g \in \mathcal{F}_{data}$ and any projection clause of a data constructor is not selection free, we deduce that there exist $\mathcal{D}_{M_1}, \ldots, \mathcal{D}_{M_\ell}$ and $\tau_{M_1}, \ldots, \tau_{M_\ell}$ such that the derivation $\mathcal{D}_{j-1}$ is of the following form:

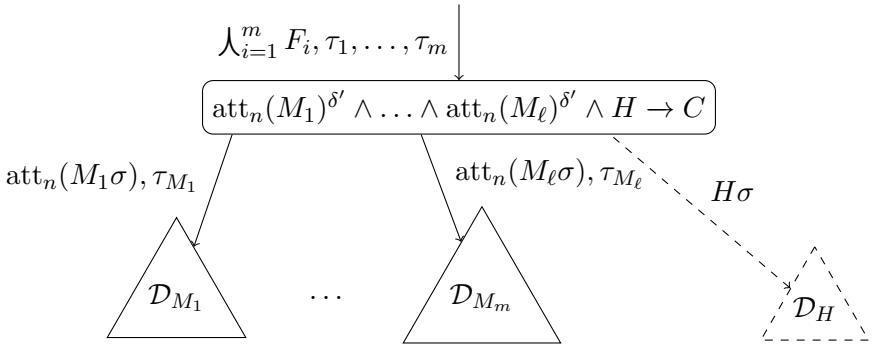


Note that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ implies that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{M_k}$ for all $k \in \{1, \ldots, \ell\}$. Furthermore, Definition 26 also indicates that for all $i \in \{1, \ldots, m\}$, if $\delta(i)$ is defined then $\tau' \ \delta(i) \ \tau_i$. Furthermore, as $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$, we also deduce that if $\text{att}_n \in \mathcal{S}_p$ (resp. $\text{att}_n \notin \mathcal{S}_p$) then $\tau_{M_j} < \tau'$ (resp. $\tau_{M_j} \leq \tau'$) for all $j \in \{1, \ldots, \ell\}$. Hence, for all $i \in \{1, \ldots, m\}$, if $\delta(i)$ is defined then

– either $\text{att}_n \in \mathcal{S}_p$ and $\tau_{M_j} < \tau'\ \delta(i)\ \tau_i$ which implies that $\tau_{M_j}\ \delta^<(i)\ \tau_i$ and so $\tau_{M_j}\ \delta'(i)\ \tau_i$;
– or else $\text{att}_n \notin \mathcal{S}_p$ and $\tau_{M_j} \leq \tau'\ \delta(i)\ \tau_i$ which implies that $\tau_{M_j}\ \delta(i)\ \tau_i$ and so $\tau_{M_j}\ \delta'(i)\ \tau_i$

In both cases, we can thus build $\mathcal{D}_j$ by replacing the ordered clause R' with R'' and by directly plugging $\mathcal{D}_{M_1}, \ldots, \mathcal{D}_{M_m}$ as child of the child of the root as follows:

The derivation $\mathcal{D}_j$



Notice that if $\text{att}_n \notin \mathcal{S}_p$ then $\delta' = \delta$. Hence, since R' satisfies $\mathcal{S}_p$, we deduce that the clause R'' also satisfies $\mathcal{S}_p$.

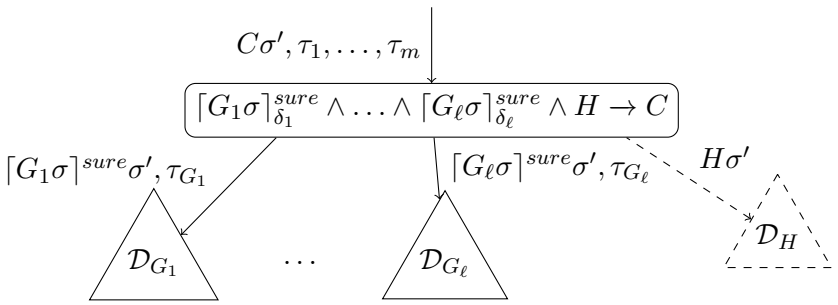
- Rule Nat_o and $\text{R}_{\phi,o}$: These sets of rules only modifies the formula of the ordered clause and preserve their solution. Hence, the result directly holds.

- Rule $\text{Lem}_o(\mathcal{L}, \mathcal{S}_p)$ and $\text{Ind}_o(\mathcal{L}_i, \mathcal{S}_p)$: We prove these two cases at the same time since main difference between these rules are on the application conditions of the rules.

  For both rules, there exist a substitution σ and $\psi = (\bigwedge_{k=1}^{\ell} G_k \Rightarrow \bigvee_{k=1}^{n} \phi_k) \in \mathcal{L}$ such that $R' = (\lceil G_1\sigma \rceil_{\delta_1}^{sure} \wedge \ldots \lceil G_\ell\sigma \rceil_{\delta_\ell}^{sure} \wedge H \to C)$ and for all $k \in \{1, \ldots, \ell\}$, $pred(G_k) \in \mathcal{S}_p$. Moreover, in the case of rule $\text{Ind}_o(\mathcal{L}_i, \mathcal{S}_p)$, we know that $\delta_1, \ldots, \delta_\ell$ are n-strict. Finally, the resulting set of clauses $\mathbb{C}'$ is $\{\lceil G_1\sigma \rceil_{\delta_1}^{sure} \wedge \ldots \lceil G_\ell\sigma \rceil_{\delta_\ell}^{sure} \wedge H \wedge \lceil \phi_k\sigma \rceil_{\delta}^{sure} \to C\}_{k=1}^{n}$ where $\delta \sqsupseteq_o \delta_k$ for all $k \in \{1, \ldots, \ell\}$. Notice that the generated clause only adds events in the hypotheses of R'. Hence, since $\mathcal{S}_p$ contains the event predicates and R' satisfies $\mathcal{S}_p$, we deduce that for all $k \in \{1, \ldots, n\}$, the clause $\lceil G_1\sigma \rceil_{\delta_1}^{sure} \wedge \ldots \lceil G_\ell\sigma \rceil_{\delta_\ell}^{sure} \wedge H \wedge \lceil \phi_k\sigma \rceil_{\delta}^{sure} \to C$ satisfies $\mathcal{S}_p$.

  We obtain that $\mathcal{D}_{j-1}$ is the derivation presented below:

The derivation $\mathcal{D}_{j-1}$



Since for all $k \in \{1,\ldots,\ell\}$, $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{G_k}$ and $pred(G_k) \in \mathcal{S}_p$, we deduce that $T, \tau_{G_k} \vdash_{IO}^{n_{IO}} \lceil G_k\sigma \rceil^{sure}\sigma'$. Recall that the satisfaction conditions for an event are the same for its sure-event counterpart. Therefore, we deduce that for all $k \in \{1,\ldots,\ell\}$, $T, \tau_{G_k} \vdash_{IO}^{n_{IO}} G_k\sigma\sigma'$.

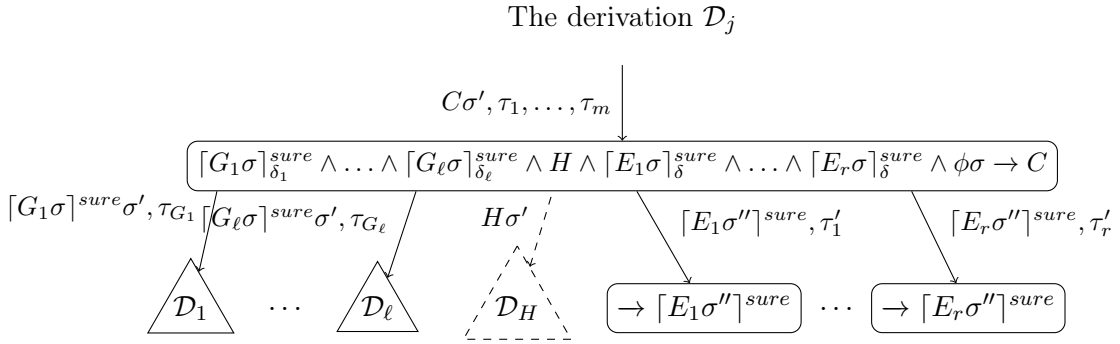
Note that when the rule applied is $\mathrm{Ind}_o(\mathcal{L}_i, \mathcal{S}_p)$, we know that $\delta_1,\ldots,\delta_\ell$ are n-strict. Moreover since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{j-1}$, we deduce that for all $k \in \{1,\ldots,\ell\}$, for $i \in \{1,\ldots,m\}$, if $\delta_k(i)$ is defined then $\tau_{G_k}\delta_k(i)\tau_i$. Therefore by Lemma 14, we deduce that $\{\!\{\tau_{G_1},\ldots,\tau_{G_\ell}\}\!\} <_m \{\!\{\tau_1,\ldots,\tau_m\}\!\}$. Hence by the hypotheses of our lemma, we obtain that $\{\!\{\tau_{G_1},\ldots,\tau_{G_\ell}\}\!\} <_m \mathcal{M}$.

This allows us to apply Lemma 29, thus there exist $i \in \{1,\ldots,n\}$, a substitution σ'', steps $\tau'_1,\ldots,\tau'_r$, events $E_1,\ldots,E_r$ and a formula ϕ' such that

- $\phi_i = E_1 \wedge \ldots \wedge E_r \wedge \phi'$
- for all $i' \in \{1,\ldots,\ell\}$, $G_{i'}\sigma\sigma' = G_{i'}\sigma''$
- for all $k \in \{1,\ldots,r\}$, $\tau'_k \leq \max(\tau_{G_1},\ldots,\tau_{G_\ell})$ and $T, \tau'_k \vdash_{IO}^{n_{IO}} E_k\sigma''$
- $\vdash_{IO}^{n_{IO}} \phi'\sigma''$

Let us prove that for all $i' \in \{1,\ldots,m\}$, if $\delta(i')$ is defined then for all $k \in \{1,\ldots,r\}$, $\tau'_k\ \delta(i')\ \tau_{i'}$. Let $i' \in \{1,\ldots,m\}$ and let us assume that $\delta(i')$ is defined. By definition of the transformation rule $\mathrm{Lem}_o(\mathcal{L}, \mathcal{S}_p)$ and $\mathrm{Ind}_o(\mathcal{L}, \mathcal{S}_p)$, we know that for all $k' \in \{1,\ldots,\ell\}$, $\delta \sqsupseteq_o \delta_{k'}$. Hence by Definition 27, $\delta_{k'}(i')$ is also defined and $\tau\ \delta_{k'}(i')\ \tau'$ implies $\tau\ \delta(i')\ \tau'$ for all steps τ, τ'. But $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{j-1}$ and $\delta_{k'}(i')$ being defined implies that $\tau_{G_{k'}}\delta_{k'}(i')\tau_{i'}$ and so $\tau_{G_{k'}}\ \delta(i')\ \tau_{i'}$. Since $\tau'_{k'} \leq \max(\tau_{G_1},\ldots,\tau_{G_\ell})$, we can conclude that $\tau'_{k'}\ \delta(i)\ \tau_{i'}$.

We can now build the derivation $\mathcal{D}'_R$ by replacing the clause R' with the clause $\lceil G_1\sigma \rceil^{sure} \wedge \ldots \wedge \lceil G_\ell\sigma \rceil^{sure} \wedge H \wedge \lceil E_1\sigma \rceil^{sure} \wedge \ldots \wedge \lceil E_r\sigma \rceil^{sure} \wedge \phi\sigma \rightarrow C$ as follows:



Finally, all E_i are events hence do not influence the size of the derivation meaning that $|\mathcal{D}_j| = |\mathcal{D}_{j-1}|$. Since all clauses $\rightarrow \lceil E_i\sigma'' \rceil^{sure}$ for $i = 1,\ldots,r$ are in $\mathbb{C}_e(T)$, we can conclude. □

Corollary 7. *Let $\mathbb{C}$ be a set of ordered clauses satisfying $\mathcal{S}_p$. Let $\mathcal{D}$ be an ordered derivation of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1,\ldots,\tau_m$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $\{\!\{\tau_1,\ldots,\tau_m\}\!\} \leq_m \mathcal{M}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. All clauses in $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ satisfy $\mathcal{S}_p$ and there exists an ordered derivation*

$\mathcal{D}'$ *of* $\bigwedge_{i=1}^{m} F_i$ *at steps* $\tau_1, \ldots, \tau_m$ *from* $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C})$ *and* $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ *such that* $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ *and* $|\mathcal{D}'| \leq |\mathcal{D}|$.

Proof. Since $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_1, \ldots, R_n\}) = \mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_1\}) \cup \ldots \cup \mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R_n\})$, the result directly follows from Lemma 36. □

D.2 Soundness of $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C})$

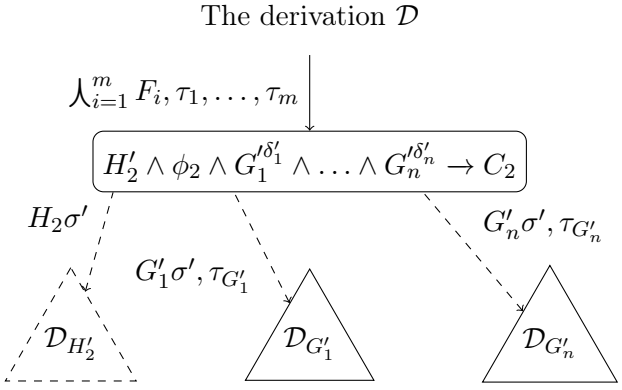
Lemma 37. *Let* $\mathcal{D}$ *be an ordered derivation of* $\bigwedge_{i=1}^{m} F_i$ *at steps* $\tau_1, \ldots, \tau_m$ *from* $\{R\}$ *and* $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ *such that* $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$.

If there exists an ordered clause R' *such that* $R' \sqsupseteq_o R$ *then there exists an ordered derivation* $\mathcal{D}'$ *of* $\bigwedge_{i=1}^{m} F_i$ *at steps* $\tau_1, \ldots, \tau_m$ *from* $\mathsf{simplify}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R'\})$ *and* $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ *such that* $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ *and* $|\mathcal{D}'| \leq |\mathcal{D}|$.

Proof. Assume that $R' \sqsupseteq_o R$. By definition, we can denote $R = H_2 \wedge H'_2 \wedge \phi_2 \rightarrow C_2$ and $R' = H_1 \wedge \phi_1 \rightarrow C_1$ such that $H_1 = \{\!\{G_1^{\delta_1}, \ldots, G_n^{\delta_n}\}\!\}$, $H_2 = \{\!\{G_1'^{\delta'_1}, \ldots, G_n'^{\delta'_n}\}\!\}$ and there exists a substitution σ such that:

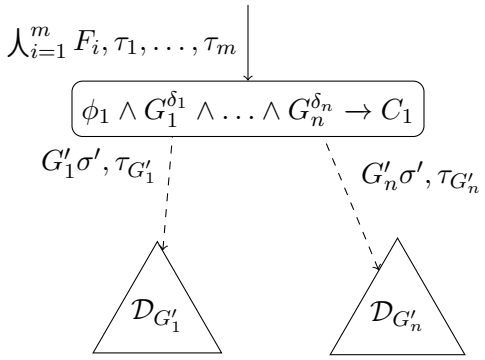
- $C_1\sigma = C_2$;
- for all $i \in \{1, \ldots, n\}$, $G_i\sigma = G'_i$ and $\delta_i \sqsupseteq_o \delta'_i$;
- $\phi_2 \models \phi_1\sigma$.

Moreover, since $\mathcal{D}$ is an ordered derivation from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$, there exist a conjunction of derivations $\mathcal{D}_{H'_2}$ and n derivations $\mathcal{D}_{G'_1}, \ldots, \mathcal{D}_{G'_n}$ and a substitution σ' such that $C_2\sigma' = \bigwedge_{i=1}^{m} F_i$, $\mathcal{D}_{H'_2}$ is the derivation of $H'_2\sigma'$ and for $i = 1 \ldots n$, $\mathcal{D}_{G'_i}$ is the derivation of $G'_i\sigma'$ at some step $\tau_{G'_i}$.



Note that $C_2\sigma' = C_1\sigma\sigma'$ and for all $i \in \{1, \ldots, n\}$, $G_i\sigma\sigma' = G'_i\sigma'$. Moreover, $\vdash_{IO}^{n_{IO}} \phi_1\sigma\sigma'$ and $\phi_2 \models \phi_1\sigma$ implies $\vdash_i \phi_1\sigma\sigma'$. Hence, we build the derivation $\mathcal{D}'$ by replacing R with R' as root and by only keeping the derivations $\mathcal{D}_{G'_1}, \ldots, \mathcal{D}_{G'_n}$ as follows:

The derivation $\mathcal{D}'$



To prove that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$, we still need to ensure that Item 3c of Definition 26 is verified, i.e. that the derivation satisfies the ordering functions $\delta_1, \ldots, \delta_n$. The other properties of Definition 26 being directly obtained from $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$.

Let $k \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m\}$. Assume that $\delta_k(i)$ is defined. We know that $\delta_k \sqsupseteq_o \delta'_k$. Hence $\delta_k(i)$ being defined implies that $\delta'_k(i)$ is defined. Hence, $\tau_{G'_k}\ \delta'_k(i)\ \tau_i$. However, from $\delta_k \sqsupseteq_o \delta'_k$, we also know that $\delta'_k(i) = \leq$ implies $\delta_k(i) \neq <$. Thus, $\delta'_k(i) = \leq$ implies $\delta_k(i) = \leq$; and $\delta'_k(i) = <$ implies $\delta_k(i) \in \{\leq, <\}$. In both cases, we deduce that $\tau_{G'_k}\ \delta'_k(i)\ \tau_i$ implies $\tau_{G'_k}\ \delta_k(i)\ \tau_i$, which allows us to conclude. □

Corollary 8. *Let $\mathbb{C} = \mathbb{C}' \cup \{R; R'\}$ be a set of ordered clauses such that $R' \sqsupseteq_o R$. Let $\mathcal{D}$ be an ordered derivation of $\curlywedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. There exists an ordered derivation $\mathcal{D}'$ of $\curlywedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}' \cup \{R'\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$.*

Proof. If the child of the root of derivation $\mathcal{D}$ is labeled by R, we can apply Lemma 37 to obtain the result. Otherwise, the clause labeling the child of the root of $\mathcal{D}$ is in $\mathbb{C}' \cup \{R'\}$ and so the result directly holds. □

Lemma 38. *Let $\mathbb{C} = \mathbb{C}' \cup \{R = (H \rightarrow C)\}$ be a set of ordered clauses satisfying $\mathcal{S}_p$ such that $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}) = \mathbb{C}$. Let $\mathcal{D}$ be an ordered derivation of $\curlywedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$.*

If there exists $\mathcal{D}_p$ an ordered partial derivation of C from $\mathbb{C}'$ and $\mathbb{C}_{sat}$ such that $\mathbb{F}_{us}(\mathcal{D}_p) \rightarrow C \sqsupseteq R$ and $pred(\mathbb{F}_s(\mathcal{D}_p)) \cap \mathcal{S}_p = \emptyset$ then there exists a derivation $\mathcal{D}'$ of $\curlywedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}'$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and either $\mathcal{D}'$ is selection free or $\mathcal{D}' = \mathcal{D}$.

Proof. Let's assume that the child of the root of $\mathcal{D}$ is the rule R, otherwise the result trivially holds by taking $\mathcal{D}' = \mathcal{D}$. By definition of $\mathcal{D}_p$ being an ordered partial derivation of C from $\mathbb{C}'$ and $\mathbb{C}_{sat}$, we deduce that the child of the root of $\mathcal{D}_p$ is labeled by an ordered clause $R' = U_1^{\delta_1} \wedge \ldots \wedge U_n^{\delta_n} \wedge \phi_u \rightarrow C'$ and there exist a substitution σ and n partial derivations $\mathcal{D}_p^1, \ldots, \mathcal{D}_p^n$ of $U_1\sigma, \ldots, U_n\sigma$ from $\mathbb{C}_{sat}$ respectively such that $C'\sigma = C$.

Moreover, we have $\mathbb{F}_{us}(\mathcal{D}_p) = \{\!\{\phi_u\sigma\}\!\} \cup \bigcup_{i=1}^{n} \mathbb{F}_i$ where for all $i \in \{1, \ldots, n\}$,

- if $\mathcal{D}_p^i$ only contains a root and an unlabeled leaf, then $\mathbb{F}_i = \{\!\{U_i^{\delta_i}\sigma\}\!\}$;
- otherwise $\mathbb{F}_{us}(\mathcal{D}_p^i)$ is some multiset $\{\!\{G_{1,i}, \ldots, G_{n_i,i}, \phi_i\}\!\}$ and $\mathbb{F}_i = \{\!\{G_{1,i}^{\delta_{G_{1,i}}}, \ldots, G_{n_i,i}^{\delta_{G_{n_i,i}}}, \phi_i\}\!\}$ where for all $j \in \{1, \ldots, n_i\}$, if $pred(G_{j,i}) \in \mathcal{S}_p$ then $\delta_{G_{j,i}} = \delta_i^{<}$ else $\delta_{G_{j,i}} = \delta_i$.

For uniformity, in the first case, we define $n_i = 1$, $G_{1,i} = U_i\sigma$, $\delta_{G_{1,i}} = \delta_i$, and $\phi_i = \top$.

By hypothesis, $\mathbb{F}_{us}(\mathcal{D}_p) \to C \sqsupseteq R$. Hence, there exists a substitution σ' such that :

- H is of the form $H' \wedge \phi' \wedge \bigwedge_{i=1}^{n} G_{1,i}^{\delta_{1,i}}\sigma' \wedge \ldots \wedge G_{n_i,i}^{\delta_{n_i,i}}\sigma'$
- $C\sigma' = C$
- $\phi' \models \phi_u\sigma\sigma' \wedge \phi_1\sigma' \wedge \ldots \wedge \phi_n\sigma'$
- $\delta_{G_{j,i}} \sqsupseteq_o \delta_{j,i}$

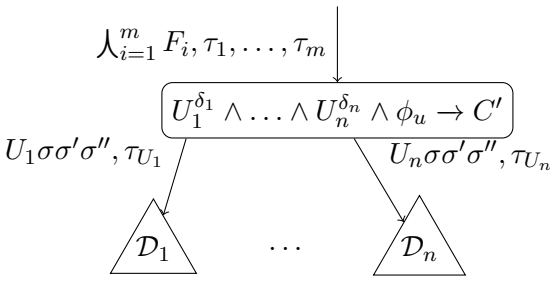
Finally, since $\mathcal{D}$ is an ordered derivation of $\curlywedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$, we obtain that there exist a substitution σ'', some steps $\tau_{j,i}$ and derivations $\mathcal{D}_{j,i}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$ such that for all $i \in \{1, \ldots, n\}$, for all $j \in \{1, \ldots, n_i\}$,

- $\mathcal{D}_{j,i}$ is a derivation of $G_{j,i}\sigma'\sigma''$ at step $\tau_{j,i}$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$
- $T, \mathcal{S}_p, n_{IO} \vdash_{IO}^{n_{IO}} \mathcal{D}_{i,j}$
- for all $k \in \{1, \ldots, m\}$, if $\delta_{j,i}(k)$ is defined then $\tau_{j,i}\ \delta_{j,i}(k)\ \tau_k$
- $\vdash_{IO}^{n_{IO}} \phi'\sigma''$
- $C\sigma'' = \curlywedge_{i=1}^{m} F_i$

Since $\phi' \models \phi_u\sigma' \wedge \phi_1\sigma' \wedge \ldots \wedge \phi_n\sigma'$ and $\vdash_{IO}^{n_{IO}} \phi'\sigma''$, we deduce that $\vdash_{IO}^{n_{IO}} \phi_u\sigma\sigma'\sigma''$ and for all $i \in \{1, \ldots, n\}$, $\vdash_{IO}^{n_{IO}} \phi_i\sigma'\sigma''$.

Let $i \in \{1, \ldots, n\}$. Let us prove that there exists a derivation $\mathcal{D}_i$ of $U_i\sigma\sigma'\sigma''$ at some step τ_{U_i} from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash_{IO}^{n_{IO}} \mathcal{D}_i$ and for all $k \in \{1, \ldots, m\}$, if $\delta_i(k)$ is defined then $\tau_{U_i}\ \delta_i(k)\ \tau_k$. With this property and since we already proved that $\vdash_{IO}^{n_{IO}} \phi_u\sigma\sigma'\sigma''$ and $C'\sigma\sigma'\sigma'' = C\sigma'\sigma'' = C\sigma'' = \curlywedge_{i=1}^{m} F_i$, we would obtain an ordered derivation $\mathcal{D}'$ from $\mathbb{C}'$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. In particular, $\mathcal{D}'$ would have the following form:

The derivation $\mathcal{D}'$



By hypothesis of the rule, we know that $\mathbb{F}_s(\mathcal{D}_p) \cap \mathcal{S}_p = \emptyset$. Let us do a case analysis on whether $pred(U_i) \in \mathcal{S}_p$ or not.

Case $pred(U_i) \in \mathcal{S}_p$: In such a case, we deduce that $\mathcal{D}_p^i$ is in fact a partial derivation composed of only the unlabeled root and the unlabeled leaf linked by an edge labeled $U_i\sigma$. Hence $n_i = 1$, $G_{1,i} = U_i\sigma$ and $\delta_{G_{1,i}} = \delta_i$. In such a case, we take $\tau_{U_i} = \tau_{1,i}$ and $\mathcal{D}_i = \mathcal{D}_{1,i}$. Indeed, we already know that $\mathcal{D}_{1,i}$ is a derivation of $G_{1,i}\sigma'\sigma''$ at step $\tau_{1,i}$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash_{IO}^{n_{IO}} \mathcal{D}_{1,i}$. Furthermore, for all $k \in \{1, \ldots, m\}$, if $\delta_i(k)$ is defined then we know

from $\delta_i = \delta_{G_{1,i}} \sqsupseteq_o \delta_{1,i}$ that $\delta_{1,i}(k)$ is defined. Hence, from $T, \mathcal{S}_p, n_{IO} \vdash^{n_{IO}}_{IO} \mathcal{D}_{1,i}$, we deduce that $\tau_{1,i}\ \delta_{1,i}(k)\ \tau_k$ which implies $\tau_{1,i}\ \delta_i(k)\ \tau_k$ thanks to $\delta_{G_{1,i}} \sqsupseteq_o \delta_{1,i}$.

Case $pred(U_i) \notin \mathcal{S}_p$: In such a case, we define $\tau_{U_i} = \max_{j=1}^{n_i}(\text{if } pred(G_{j,i}) \in \mathcal{S}_p \text{ then } \tau_{j,i} + 1 \text{ else } \tau_{j,i})$. We wish to apply Lemma 34. We already know that $\mathcal{D}^i_p$ is a partial derivation of $U_i\sigma$ where $\mathbb{F}_{us}(\mathcal{D}^i_p) = \{\!\{G_{j,i}, \ldots, G_{n_i,i}, \phi_i\}\!\}$. Furthermore, we also proved that $\vdash^{n_{IO}}_{IO} \phi_i\sigma'\sigma''$ and for all $j \in \{1, \ldots, n_i\}$, $\mathcal{D}_{j,i}$ is a derivation of $G_{j,i}\sigma'\sigma''$ at step $\tau_{j,i}$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_{j,i}$, hence satisfying the first and third item of Lemma 34's hypotheses. Since $pred(U_i) \notin \mathcal{S}_p$, the fourth item of Lemma 34's hypotheses is trivially satisfied. Finally, the second item of Lemma 34's hypotheses is satisfied by construction of τ_{U_i}. Therefore, we can apply Lemma 34 to obtain that there exists a derivation $\mathcal{D}_i$ of $U_i\sigma\sigma'\sigma''$ at step τ_{U_i} from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$.

It remains to prove that for all $k \in \{1, \ldots, m\}$, if $\delta_i(k)$ is defined then $\tau_{U_i}\ \delta_i(k)\ \tau_k$. Let us consider $j \in \{1, \ldots, n_i\}$ such that $\tau_{U_i} = \tau_{j,i} + 1$ when $pred(G_{j,i}) \in \mathcal{S}_p$ and $\tau_{U_i} = \tau_{j,i}$ when $pred(G_{j,i}) \notin \mathcal{S}_p$.

- Case $pred(G_{j,i}) \in \mathcal{S}_p$ and $\tau_{U_i} = \tau_{j,i} + 1$: In such a case, $\delta_{G_{j,i}} = \delta^<_i$. Hence, $\delta_i(k)$ being defined implies $\delta_{G_{j,i}}(k) = <$. Moreover, since $\delta_{G_{j,i}} \sqsupseteq_o \delta_{j,i}$, we deduce that $\delta_{j,i}(k) = <$. But we already proved that in such case, $\tau_{j,i}\ \delta_{j,i}(k)\ \tau_k$, i.e. $\tau_{j,i} < \tau_k$, and so $\tau_{U_i} \leq \tau_k$. Note that that since $pred(U_i) \notin \mathcal{S}_p$ and R' is an ordered clause satisfying $\mathcal{S}_p$, we deduce that $\delta_i(k) = \leq$. Thus we conclude that $\tau_{U_i}\ \delta_i(k)\ \tau_k$.

- Case $pred(G_{j,i}) \notin \mathcal{S}_p$ and $\tau_{U_i} = \tau_{j,i}$: In such a case, $\delta_{G_{j,i}} = \delta_i$. Moreover, since $\delta_{G_{j,i}} \sqsupseteq_o \delta_{j,i}$, we deduce that $\delta_i \sqsupseteq_o \delta_{j,i}$ and so $\delta_i(k)$ being defined implies that $\delta_{j,i}(k)$ is defined. Since $\tau_{j,i}\ \delta_{j,i}(k)\ \tau_k$ and $\delta_i \sqsupseteq_o \delta_{j,i}$, we obtain that $\tau_{j,i}\ \delta_i(k)\ \tau_k$ and so we can conclude that $\tau_{U_i}\ \delta_i(k)\ \tau_k$. □

Corollary 9. *Let $\mathbb{C}$ be a set of ordered clauses satisfying $\mathcal{S}_p$ such that $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\mathbb{C}) = \mathbb{C}$. Let $\mathcal{D}$ be an ordered derivation of $\curlywedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. All clauses in $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C})$ satisfy $\mathcal{S}_p$ and there exists a derivation $\mathcal{D}'$ of $\curlywedge^m_{i=1} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C})$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$.*

Proof. $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C}) \subseteq \mathbb{C}$ hence we directly have that all clauses in $\mathsf{condenseS}_{\mathcal{S}_p}(\mathbb{C})$ satisfy $\mathcal{S}_p$. The rest of the proof is a direct application of Corollary 8 and lemma 38. □

D.3 Main proof

Theorem 4. *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{S}_p$ be a set of predicates containing the event predicates* m-event *and* s-event*. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of lemmas. let $\mathcal{R}$ be a set of fully IO-κ_{io}-compliant lemmas. Let $T \in \mathsf{trace}^{\kappa_{io}}_{IO}(\mathcal{C}_I, \rightarrow_i)_{|\mathcal{R}}$. Let $\mathbb{C}$ be a set of well originated, simplified, selection free clauses containing the selection free clauses of $\mathbb{C}_{std}$.*

For all ordered clauses R satisfying $\mathcal{S}_p$, for all ordered derivations $\mathcal{D}$ of $\curlywedge^m_{i=1} F_i$ at steps $\tilde{\tau}$ from $\{R\}$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ and $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}$, there exists an ordered derivation $\mathcal{D}'$ of $\curlywedge^m_{i=1} F_i$ at steps $\tilde{\tau}$ from $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{R},\mathcal{L}_i}(\{R\}, \mathbb{C})$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, \kappa_{io} \vdash \mathcal{D}'$.

Proof. The first part of the proof is almost a direct consequence of Corollaries 7 and 9.

The first step of $\mathsf{saturateS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}, \mathbb{C})$ consists of generating $\mathbb{C}_1 = \mathsf{condenseS}_{\mathcal{S}_p}(\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}))$. Since $\{\!\{\tau_1, \ldots, \tau_m\}\!\} \leq_{ind} \mathcal{M}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$, we can apply Corollaries 7 and 9 to obtain

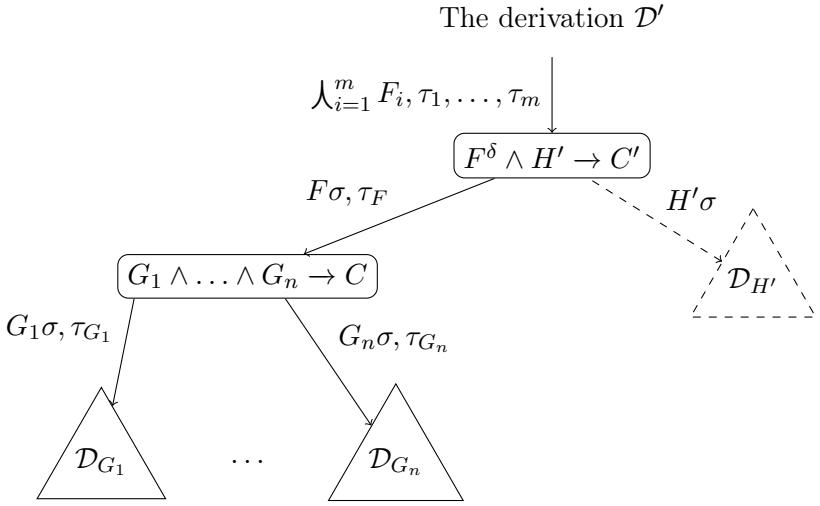
that there exists a derivation $\mathcal{D}_1$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_1$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}_1$. The second step of the saturation generates a new ordered clause R by application of the resolution rule $\text{Res}_o(\mathcal{S}_p)$. Note that since ordered clauses in $\mathbb{C}_1$ satisfy $\mathcal{S}_p$, the clause R also satisfies $\mathcal{S}_p$. Hence $\mathcal{D}_1$ is a derivation from $\mathbb{C}_1 \cup \{R\}$ and $\mathbb{C} \cup \mathbb{C}_e(T)$. The third step of the saturation computes the set $\mathbb{C}_2 = \mathsf{condense}\mathsf{S}_{\mathcal{S}_p}(\mathbb{C}_1 \cup \mathsf{simplify}\mathsf{S}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}))$. Thus by applying Corollaries 7 and 9, we obtain that there exists a derivation $\mathcal{D}_2$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_2$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}_2$. Since the second and third steps are repeated until a fix point is reached, denoted $\mathbb{C}_N$, we can deduce that there exists a derivation $\mathcal{D}_N$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_N$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}_N$.

By definition, $\mathsf{saturate}\mathsf{S}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R\}, \mathbb{C}) = \{R \in \mathbb{C}_N \mid \text{sel}(R) = \emptyset\}$. Hence, we still have to show that we can build a selection free derivation $\mathcal{D}'$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_N$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n \vdash \mathcal{D}'$.

By contradiction, let us assume that there is no possible selection free derivation $\mathcal{D}'$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_N$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. Hence let us consider the derivation $\mathcal{D}'$ of $\bigwedge_{i=1}^{m} F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathbb{C}_N$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ that is minimal for $|\mathcal{D}'|$.

By hypothesis of the theorem, all clauses of $\mathbb{C}$ are selection free, so the ordered clause labeling the child of the root of $\mathcal{D}'$ is not selection free. Let us denote R' such ordered clause and so $\mathbb{C}_N = \mathbb{C}'_N \cup \{R'\}$. Since R' is not selection free, we deduce that $R' = F^{\delta} \wedge H' \rightarrow C'$ for some F, δ, H', C' such that $F \in \text{sel}(R')$.

By definition of a derivation, we deduce that $\mathcal{D}'$ is of the following form:



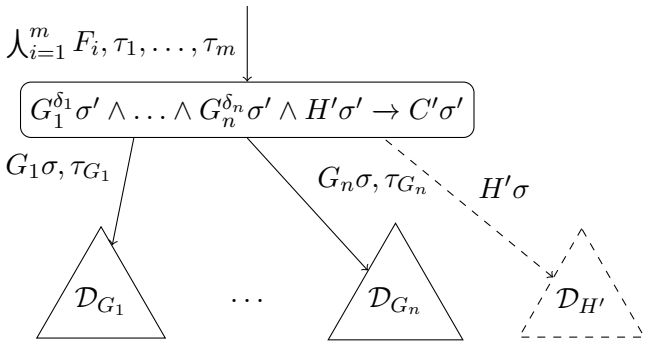
where σ is a substitution, $\mathcal{D}_{H'}$ is a conjunction derivation of $H'\sigma$ and for all $i \in \{1, \ldots, n\}$, $\mathcal{D}_{G_i}$ is a derivation of $G_i\sigma$ at step τ_{G_i}. Note that since F is selectable, F cannot be an event. Thus $(G_1 \wedge \ldots \wedge G_n \rightarrow C) \notin \mathbb{C}_e(T)$ and so $(G_1 \wedge \ldots \wedge G_n \rightarrow C) \in \mathbb{C}$ and so the clause $G_1 \wedge \ldots \wedge G_n \rightarrow C$ is selection free. Moreover since $F\sigma$ is the label of the incoming edge of the node labeled by $G_1 \wedge \ldots \wedge G_n \rightarrow C$, we deduce that $C\sigma = F\sigma$. Thus, we can compute σ' the most general unifier of C and F and apply the rule $\text{Res}_o(\mathcal{S}_p)$ to obtain the clause $R'' = G_1^{\delta_1}\sigma' \wedge \ldots \wedge G_n^{\delta_n} \wedge H'\sigma' \rightarrow C'\sigma'$ where for all $i \in \{1, \ldots, n\}$, if $pred(G_i) \in \mathcal{S}_p$ then $\delta_i = \delta^{<}$ else $\delta_i = \delta$.

Note that since $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$, we deduce that for all $i \in \{1, \ldots, n\}$, if $pred(G_i) \in \mathcal{S}_p$ then $\tau_{G_i} < \tau_F$ else $\tau_{G_i} \leq \tau_F$. Moreover, for all $k \in \{1, \ldots, m\}$, if $\delta(k)$ is defined then $\tau_F \ \delta(k) \ \tau_k$.

Therefore, let $i \in \{1, \ldots, n\}$, let $k \in \{1, \ldots, m\}$ and assume that $\delta_i(k)$ is defined. We show that $\tau_{G_i}\ \delta_i(k)\ \tau_k$. If $pred(G_i) \in \mathcal{S}_p$ then by definition $\delta_i = \delta^<$. Hence, we deduce that $\delta_i(k) = <$. But $\delta_i(k)$ being defined implies $\delta(k)$ is defined and so $\tau_F\ \delta(k)\ \tau_k$. With $\tau_{G_i} < \tau_F$, we deduce that $\tau_{G_i} < \tau_k$ which allows us to conclude. Otherwise $pred(G_i) \notin \mathcal{S}_p$ and so by definition $\delta_i = \delta$ and so $\tau_F\ \delta_i(k)\ \tau_k$. Since $\tau_{G_i} \leq \tau_F$, we conclude that $\tau_{G_i}\ \delta_i(k)\ \tau_k$.

Thus we can build the following derivation $\mathcal{D}'_1$ of $\bigwedge_{i=1}^m F_i$ at steps $\tau_1, \ldots, \tau_m$:

The derivation $\mathcal{D}'_1$



Notice that $|\mathcal{D}'_1| < |\mathcal{D}'|$. By applying Lemma 36, we obtain that there exists another ordered derivation $\mathcal{D}'_2$ of $\bigwedge_{i=1}^m F_i$ at steps $\tau_1, \ldots, \tau_m$ from $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R''\})$ and $\mathbb{C} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_2$ and $|\mathcal{D}'_2| \leq |\mathcal{D}'_1| < |\mathcal{D}'|$.

However, $\mathbb{C}_N$ was obtained by applying the saturation steps until a fix point is reached. Hence by our minimality argument, the ordered clause from $\mathsf{simplifyS}^{\mathcal{S}_p}_{\mathcal{L},\mathcal{L}_i}(\{R''\})$ labeling the child of the root of $\mathcal{D}'_2$ should not occur in $\mathbb{C}_N$ and so must have been removed by either an application of the subsumption rule or the rule $(\text{GRed}_o(\mathcal{S}_p))$. But applying Lemmas 37 and 38, we either obtain a selection free derivation of $\bigwedge_{i=1}^m F_i$ at steps $\tau_1, \ldots, \tau_m$ or we obtain a derivation $\mathcal{D}'_3$ of $\bigwedge_{i=1}^m F_i$ at steps $\tau_1, \ldots, \tau_m$ such that $|\mathcal{D}'_3| \leq |\mathcal{D}'_2|$ which implies $|\mathcal{D}'_3| < |\mathcal{D}'|$. In both cases, it contradicts our minimality hypothesis which allows us to conclude. □

E Proof of Theorem 6

In this section, we consider the following preamble $\mathcal{P}_{\mathsf{verif}}$:

> *Let $\mathcal{C}_I$ be an initial instrumented configuration. Let $\mathcal{Q}$ be a set of IO-n_{IO}-compliant queries. Let $\mathcal{L}$, $\mathcal{L}_i$ be two sets of* ProVerif *IO-n_{IO}-compliant lemmas. Let $\mathcal{S}_p$ be a set of predicates containing the event predicates* m-event *and* s-event*; all predicates in the conclusion of queries in $\mathcal{Q}$; all predicates in $\mathcal{L}$ and $\mathcal{L}_i$ and such that for all $i \in \mathbb{N}$, if* $\text{att}_i \in \mathcal{S}_p$ *(resp.* table_i*) then for all* $0 \leq j \leq i$, $\text{att}_j \in \mathcal{S}_p$ *(resp.* $\text{table}_j \in \mathcal{S}_p$*).*

E.1 Preliminary lemmas

Lemma 39. *Let $\mathcal{C}_I$ be an initial instrumented configuration, let n_{IO} be an integer and $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$. For all tuples of integers $\tilde{\tau}, \tilde{\tau}'$, if $\tilde{\tau}' \leq_m \tilde{\tau}$ and $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ then $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau}')$.*

Proof. The proof almost follows immediately from Definition 22. The first item of Definition 22 for $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau}')$ directly holds from $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$. For the second item, since $\tilde{\tau}' \leq_m \tilde{\tau}$, we have that $(T, \tilde{\tau}') \leq_{ind} (T, \tilde{\tau})$. Hence, for all $T' \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \to_i)$, for all tuples of steps

$\tilde{\tau}''$, if $(T', \tau'') <_{ind} (T, \tilde{\tau}')$ then $(T', \tau'') <_{ind} (T, \tilde{\tau})$ which implies $\mathcal{IH}_\varrho(T', \tilde{\tau}'')$ thanks to $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$. □

Lemma 40. *Consider the preamble* $\mathcal{P}_{\mathsf{verif}}$. *Let* $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. *Let* $(\mathcal{L}', \mathcal{L}'_i)$ *such that either* $\mathcal{L}' = \mathcal{L}$ *and* $\mathcal{L}'_i = \mathcal{L}_i$ *or* $\mathcal{L}' = \mathcal{L} \cup \mathcal{L}_i$ *and* $\mathcal{L}' = \emptyset$.

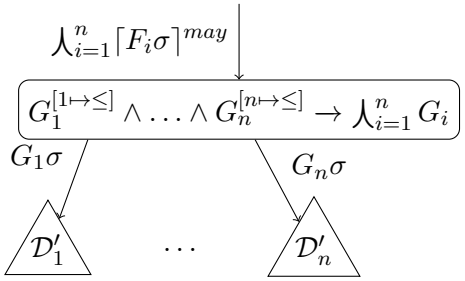
For all $\varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \psi) \in \mathcal{Q}$, *for all substitutions* σ, *for all tuples of steps* $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, *if* $\mathcal{H}yp_{\mathcal{L}',\mathcal{L}'_i}(T, \tilde{\tau})$ *and for all* $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash^{n_{IO}}_{IO} F_i\sigma$ *then there exists a derivation* $\mathcal{D}$ *of* $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ *at steps* $\tilde{\tau}$ *from* $\{R_\varrho\}$ *and* $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ *such that:*

- $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$
- $\mathbb{C}_{sat} = \mathsf{saturate}^{\mathcal{S}_p}_{\mathcal{L}',\mathcal{L}'_i}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}_{\mathcal{A}}(\mathcal{C}_I))$
- $R_\varrho = (G_1^{[1 \mapsto \leq]} \wedge \ldots \wedge G_n^{[n \mapsto \leq]} \rightarrow \bigwedge_{i=1}^{n} G_i)$ *with* $G_i = \lceil F_i \rceil^{may}$ *for* $i = 1 \ldots n$.

Proof. By definition, we know that for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash^{n_{IO}}_{IO} F_i\sigma$ implies $T, \tau_i \vdash^{n_{IO}}_{IO} \lceil F_i\sigma \rceil^{may}$. Since $\lceil F_i\sigma \rceil^{may}$ is not a sure-event, we can apply Theorem 1 to obtain that there exists a derivation $\mathcal{D}_i$ of $\lceil F_i\sigma \rceil^{may}$ at step τ_i from $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO}) \cup \mathbb{C}_e^{<\tau_i}(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_i$. Note that $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$ contains $\mathbb{C}_{std}$. Moreover, by Lemma 28, we know that $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I, n_{IO})$ is a well originated set of clauses. Since for all $i \in \{1, \ldots, n\}$, $(\tau_i) \leq_m \tilde{\tau}$, we can apply Lemma 39 to obtain that $\mathcal{H}yp_{\mathcal{L}',\mathcal{L}'_i}(T, (\tau_i))$ holds. This allows us to apply Theorem 2 and so for all $i \in \{1, \ldots, n\}$, there exists a derivation $\mathcal{D}'_i$ of $\lceil F_i\sigma \rceil^{may}$ at step τ_i from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_i$.

By definition of the saturation procedure, we know that clauses in $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ are all selection-free clauses. Thus, we can define the ordered derivation $\mathcal{D}$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at step $\tau_1, \ldots, \tau_n$ from $\{R_\varrho\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ as follows:

The derivation $\mathcal{D}$



It remains to show that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. We already showed that for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash^{n_{IO}}_{IO} \lceil F_i\sigma \rceil^{may}$ and $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_i$. Finally, for all $i \in \{1, \ldots, n\}$, we trivially have that $\tau_i\ [i \mapsto \leq](i)\ \tau_i$. This allows us to conclude that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$. □

Lemma 41. *Consider the preamble* $\mathcal{P}_{\mathsf{verif}}$. *Let* $\tau_1, \ldots, \tau_n$ *be* n *steps. Let* F^δ *be an ordered fact such that* $pred(F) \in \mathcal{S}_p$. *Let* H *be a set of pairs of ordered facts and steps. Assume that the domains of ordering functions are in* $\{1, \ldots, n\}$.

Let $H' = \{F' \mid (F'^{\delta'}, \tau') \in H \wedge \delta \sqsupseteq_o \delta'\}$. *If there exists a substitution* σ *such that:*

- *for all* $(F'^{\delta'}, \tau') \in H$, *if* $pred(F') \in \mathcal{S}_p$ *then* $T, \tau' \vdash_i F'\sigma$; *and for all* $i \in \{1, \ldots, n\}$, $\delta'(i)$ *being defined implies* $\tau'\ \delta'(i)\ \tau_i$;
- F *is deducible from* H' *under some* ϕ *such that* $\vdash_i \phi\sigma$

then there exists $\tau \leq \max(\{\tau' \mid (F'^{\delta'}, \tau') \in H \wedge \delta \sqsupseteq_o \delta'\})$ *such that* $T, \tau \vdash_i F\sigma$ *and for all* $i \in \mathrm{dom}(\delta)$, $\tau\ \delta(i)\ \tau_i$.

Proof. By definition, since F is deducible from H' under some ϕ, we know that there exists a partial derivation $\mathcal{D}$ of F from $\{(RInit), (RGen), (RFail), (Rf), (Rap), (Rtp)\}$ such that:

- all facts in $\mathbb{F}_{us}(\mathcal{D})$ are in H'
- ϕ is the conjunction of all formulae in $\mathbb{F}_{us}(\mathcal{D})$

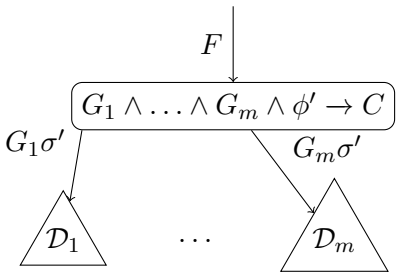
Let us prove this result by induction on the size of the partial derivation $\mathcal{D}$.

Base case, $\mathcal{D}$ *contains only a root and unlabeled leaf:* In such a case, F is in fact an element of H'. Hence, there exists δ' and τ' such that $(F^{\delta'}, \tau') \in H$ and $\delta \sqsupseteq_o \delta'$. Moreover, we also have that $T, \tau' \vdash_i F\sigma$ and for all $i \in \{1, \ldots, n\}$, $\delta'(i)$ being defined implies $\tau'\delta'(i)\tau_i$. Since $\delta \sqsupseteq_o \delta'$, we deduce that for all $i \in \mathrm{dom}(\delta)$, $\tau'\ \delta(i)\ \tau_i$. Hence the result holds by taking $\tau = \tau'$.

Inductive step, the child of the root of $\mathcal{D}$ *is a labeled node:* Let us denote by $R = (G_1 \wedge \ldots \wedge G_m \wedge \phi' \rightarrow C)$ the rule labeling the root's child of $\mathcal{D}$.

By definition of a partial derivation, we know that there exists σ' such that $C\sigma' = F$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $G_1\sigma', \ldots, G_m\sigma'$. Let us denote by $\mathcal{D}_1, \ldots, \mathcal{D}_m$ the partial derivation of $G_1\sigma', \ldots, G_m\sigma'$ respectively. Hence $\mathcal{D}$ is of the form:

The derivation $\mathcal{D}$



Note that $\mathbb{F}_{us}(\mathcal{D}) = \{\phi'\sigma'\} \cup \bigcup_{j=1}^{m} \mathbb{F}_{us}(\mathcal{D}_j)$. Therefore, if for all $j \in \{1, \ldots, m\}$, we denote by ϕ_j the conjunction of all formulae in $\mathbb{F}_{us}(\mathcal{D}_j)$ then for all $j \in \{1, \ldots, m\}$, $G_j\sigma'$ is deducible from H' under ϕ_j. Moreover, we know by hypothesis that $\vdash_i \phi\sigma$ where ϕ is the conjunction of all formulae in $\mathbb{F}_{us}(\mathcal{D})$. Therefore, we obtain that for all $j \in \{1, \ldots, m\}$, $\vdash_i \phi_j\sigma$.

Note that $R \in \{(RInit), (RGen), (RFail), (Rf), (Rap), (Rtp)\}$ and we know that $pred(C)$ is either att_k or table_k for some k. Thus by $pred(F) \in \mathcal{S}_p$ and by our preamble $\mathcal{P}_{\mathsf{verif}}$, we know that for all $j \in \{1, \ldots, m\}$, $pred(G_j) \in \mathcal{S}_p$. Since $|\mathcal{D}_j| < |\mathcal{D}|$ for all $j \in \{1, \ldots, m\}$, we can apply our inductive hypothesis to obtain that there exist $\tau'_1, \ldots, \tau'_m$ such that for all $j \in \{1, \ldots, m\}$, $\tau'_j \leq \max(\{\tau' \mid (F'^{\delta'}, \tau') \in H \wedge \delta \sqsupseteq_o \delta'\})$, $T, \tau'_j \vdash_i G_j\sigma'\sigma$ and for all $i \in \mathrm{dom}(\delta)$, $\tau'_j\ \delta(i)\ \tau_i$.

As mentioned, we know that $pred(C)$ is either att_k or table_k for some k. The satisfaction of these two predicates is monotonous, i.e. if they are satisfied at some step τ then they are satisfied for any step after τ. Thus, by choosing $\tau = \max(\tau'_1, \ldots, \tau'_m)$, we obtain that for all $j \in \{1, \ldots, m\}$, $T, \tau \vdash_i G_j\sigma'\sigma$. Finally, when $R \in \{(RInit), (RGen), (RFail), (Rf), (Rap)\}$, it corresponds to either the rule I-App, I-New or I-Phase which gives us that $T, \tau \vdash_i C\sigma'\sigma$ and so $T, \tau \vdash_i F\sigma$. When R is the rule (Rtp), then $m = 1$, $G_1\sigma' = \mathrm{table}_k(M)$ and $F = \mathrm{table}_{k+1}(M)$ for some k, M. By definition of the satisfaction relation, $T, \tau \vdash_i \mathrm{table}_k(M)\sigma$ implies $T, \tau \vdash_i \mathrm{table}_{k+1}(M)\sigma$ and so $T, \tau \vdash_i F\sigma$.

Finally, we had $\tau'_j \leq \max(\{\tau' \mid (F'^{\delta'}, \tau') \in H \wedge \delta \sqsupseteq_o \delta'\})$ for all $j \in \{1, \ldots, m\}$. With $\tau = \max(\tau'_1, \ldots, \tau'_m)$, we deduce that $\tau \leq \max(\{\tau' \mid (F'^{\delta'}, \tau') \in H \wedge \delta \sqsupseteq_o \delta'\})$. Furthermore, for all $i \in \mathrm{dom}(\delta)$, for all τ'_j $\delta(i)$ τ_i. Hence $\max(\tau'_1, \ldots, \tau'_m)$ $\delta(i)$ τ_i and so τ $\delta(i)$ τ_i. □

Lemma 42. *Consider the preamble $\mathcal{P}_{\mathsf{verif}}$. Let $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. Let $\varrho = (\bigwedge^n_{i=1} F_i \Rightarrow \psi)$ be a disjunct query. Let $(R, \sigma_s, \mathbb{M})$ be a solution of ϱ, with $R = (H \wedge \phi \rightarrow C)$. Let σ be a substitution. Let $\mathcal{D}$ be an ordered derivation of $\curlywedge^n_{i=1} \lceil F_i\sigma \rceil^{may}$ at some steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and the root's child of $\mathcal{D}$ is labeled by R.*

There exists an annotated query conclusion Ψ and a substitution σ_d such that:

- *$C = \curlywedge^n_{i=1} \lceil F_i\sigma_s \rceil^{may}$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$;*
- *$F_i\sigma_s\sigma_d = F_i\sigma$ for $i = 1 \ldots n$ and $\overline{\Psi} = \psi$;*
- *$(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_s\sigma_d$;*
- *for all $F^{\delta,\mu} \in \Psi$, $\mathrm{dom}(\mu) = \{(\tilde{\tau}, \sigma)\}$;*
- *for all $F^{\delta,\mu} \in \Psi$ with F an (inj-)event, there exists δ' such that $(F^{\delta}, \delta') \in \mathbb{M}$ such that for all $i \in \mathrm{dom}(\delta')$, $\mu(\tilde{\tau}, \sigma)$ $\delta'(i)$ τ_i.*

Proof. Let us rename H by $G_1^{\delta_1} \wedge \ldots \wedge G_m^{\delta_m}$. With $(R, \sigma_s, \mathbb{M})$ being a solution of ϱ, we deduce that $C = \curlywedge^n_{i=1} F'_i$ where $F'_i = \lceil F_i\sigma_s \rceil^{may}$ for $i = 1 \ldots n$. Moreover, since the root's child of $\mathcal{D}$ is labeled by R, we know that there exist a substitution σ_d and some steps $\tau_{G_1}, \ldots, \tau_{G_m}$ such that:

- for all $i \in \{1, \ldots, n\}$, $F'_i\sigma_d = \lceil F_i\sigma \rceil^{may}$
- $\vdash_i \phi\sigma_d$
- for all $j \in \{1, \ldots, m\}$, if $pred(G_j) \in \mathcal{S}_p$ then $T, \tau_{G_j} \vdash^{n_{IO}}_{IO} G_j\sigma_d$
- for all $j \in \{1, \ldots, m\}$, for all $i \in \{1, \ldots, n\}$, if $\delta_j(i)$ is defined then τ_{G_j} $\delta_j(i)$ τ_i.

Note that since $F'_i = \lceil F_i\sigma_s \rceil^{may}$, we obtain that $F_i\sigma_s\sigma_d = F_i\sigma$ for $i = 1 \ldots n$.

Consider a formula ϕ' in ψ. By definition of $(R, \sigma_s, \mathbb{M})$ being a solution of ϱ, we deduce that $\phi \models \phi'\sigma_s$. Since $\vdash_i \phi\sigma_d$, we obtain that $\vdash_i \phi'\sigma_s\sigma_d$.

Let us show how to build Ψ. Consider $F^{\delta} \in \psi$ an ordered event. Since $(R, \sigma_s, \mathbb{M})$ is a solution of ϱ, we know that there exists δ' such that $(F^{\delta}, \delta') \in \mathbb{M}$, $\delta \sqsupseteq_o \delta'$ and one of the following two properties holds:

1. there exists $i \in \{1, \ldots, n\}$ such that $\lceil F\sigma_s \rceil^{may} = F'_i$ and $\delta' = [i \mapsto \leq]$: In such a case, we annotate F^{δ} in Ψ by the partial function μ only defined on $(\tilde{\tau}, \sigma)$ such that $\mu(\tilde{\tau}, \sigma) = \tau_i$. Notice that thanks to $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$, we know that $T, \tau_i \vdash^{n_{IO}}_{IO} \lceil F_i\sigma \rceil^{may}$ and $F'_i\sigma_d = \lceil F_i\sigma \rceil^{may}$. Hence we obtain that $\lceil F\sigma_s\sigma_d \rceil^{may} = \lceil F_i\sigma \rceil^{may}$ and so $T, \tau_i \vdash^{n_{IO}}_{IO} \lceil F\sigma_s\sigma_d \rceil^{may}$ which implies $T, \tau_i \vdash^{n_{IO}}_{IO} F\sigma_s\sigma_d$. This allows us to deduce that $T, \tau_i \vdash_i F\sigma_s\sigma_d$.

   Consider $k \in \{1, \ldots, n\}$ such that $\delta(k)$ is defined. We know that $\delta \sqsupseteq_o \delta'$ with $\delta' = [i \mapsto \leq]$. Hence, $\delta(k)$ being defined implies $\delta'(k)$ is defined and so $k = i$. Moreover, we also deduce from $\delta \sqsupseteq_o \delta'$ that $\delta(i) = \leq$. Since we defined $\mu(\tilde{\tau}, \sigma) = \tau_i$, we trivially have that $\mu(\tilde{\tau}, \sigma)$ $\delta(i)$ τ_i.

2. there exists $j \in \{1, \ldots, m\}$ such that $\lceil F\sigma_s \rceil^{sure} = G_j$ and $\delta' = \delta_j$: In such a case, we annotate F^δ in Φ by the partial function μ only defined on $(\tilde{\tau}, \sigma)$ such that $\mu(\tilde{\tau}, \sigma) = \tau_{G_j}$. Since $\lceil F\sigma_s \rceil^{sure} = G_j$, we have $\lceil F\sigma_s\sigma_d \rceil^{sure} = G_j\sigma_d$. Since F is an ordered event and $\lceil F\sigma_s \rceil^{sure} = G_j$, then $pred(G_j) \in \mathcal{S}_p$ by our preamble $\mathcal{P}_{\mathsf{verif}}$ and so $T, \tau_{G_j} \vdash_{IO}^{n_{IO}} G_j\sigma_d$ by hypothesis. Hence, we deduce that $T, \tau_{G_j} \vdash_{IO}^{n_{IO}} \lceil F\sigma_s\sigma_d \rceil^{sure}$ which implies $T, \tau_{G_j} \vdash_{IO}^{n_{IO}} F\sigma_s\sigma_d$ and so $T, \tau_{G_j} \vdash_i F\sigma_s\sigma_d$.

   Consider $k \in \{1, \ldots, n\}$ such that $\delta(k)$ is defined. We know that $\delta \sqsupseteq_o \delta_j$. Hence $\delta(k)$ being defined implies $\delta_j(k)$ is defined. Moreover, since $\tau_{G_j}\delta_j(k)\tau_k$ by hypothesis, we deduce from $\delta \sqsupseteq_o \delta_j$ that $\tau_{G_j}\ \delta(k)\ \tau_k$.

In both cases, we have shown that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models F^{\mu,\delta}$.

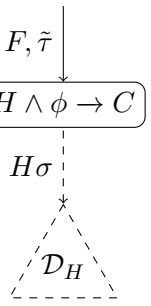
Finally, let us consider $F^\delta \in \psi$ an ordered attacker, message or table fact. Since $(R, \sigma_s, \mathbb{M})$ is a solution of ϱ, we know that $F\sigma_s$ is deducible from $\{G_j \mid \delta \sqsupseteq_o \delta_j \wedge j = 1 \ldots m \wedge pred(G_j) \in \mathcal{S}_p\}$ under some ϕ'' such that $\phi \models \phi''$. Let us consider the set $H_0 = \{(G_j^{\delta_j}, \tau_{G_j})\}_{j=1}^m$. We already showed at the beginning of this proof that $\vdash_i \phi\sigma_d$; for all $j \in \{1, \ldots, m\}$, $T, \tau_{G_j} \vdash_{IO}^{n_{IO}} G_j\sigma_d$; and for all $i \in \{1, \ldots, n\}$, if $\delta_j(i)$ is defined then $\tau_{G_j}\ \delta_j(i)\ \tau_i$. Hence by Lemma 41, there exists $\tau \leq \max(\{\tau_{G_j} \mid \delta \sqsupseteq_o \delta_j \wedge j = 1 \ldots m\})$ such that $T, \tau \vdash_i F\sigma_s\sigma_d$ and for all $i \in \mathrm{dom}(\delta)$, $\tau\ \delta(i)\ \tau_i$. Hence, we annotate F^δ in Φ by the partial function μ only defined on $(\tilde{\tau}, \sigma)$ such that $\mu(\tilde{\tau}, \sigma) = \tau$. This allows us to conclude that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_s\sigma_d$. □

Lemma 43. *Consider the preamble $\mathcal{P}_{\mathsf{verif}}$. Let $\mathbb{C}_{sat}$ be a set of selection-free clauses. Let $\mathbb{C}$ and $\mathbb{C}'$ be sets of ordered clauses such that $\mathbb{C}$ completely covers $\mathbb{C}'$.*

For all $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$, for all tuples of steps $\tilde{\tau}$, for all ground conjunction facts F, for all ordered derivations $\mathcal{D}$ of F at steps $\tilde{\tau}$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$, there exists a derivation $\mathcal{D}'$ of F at steps $\tilde{\tau}$ from $\mathbb{C}'$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$

Proof. By Definition 26, $\mathcal{D}$ being an ordered derivation of F at steps $\tilde{\tau}$ from $\mathbb{C}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ implies that there exists an ordered clause $R = (H \wedge \phi \rightarrow C) \in \mathbb{C}$ and a substitution σ such that $C\sigma = F$, $\sigma \vdash_i \phi$, $H\sigma$ is ground and $\mathcal{D}$ is as follows:

The derivation $\mathcal{D}$



By Definition 31, we know that there exist formulas $\phi_1, \ldots, \phi_m$ such that $\phi_1 \vee \ldots \vee \phi_m \equiv \top$, for all $i \in \{1, \ldots, m\}$, $fv(\phi_i) \subseteq fv(C, H, \phi)$. Considering that $\sigma \models \phi$, σ is grounding for C, H and $\phi_1 \vee \ldots \vee \phi_m \equiv \top$, we deduce that there exists $i \in \{1, \ldots, m\}$ such that $\sigma \models \phi_i$ which implies $\sigma \models \phi \wedge \phi_i$. Note that by Definition 31, the ordered clause $H \wedge \phi \wedge \phi_i \rightarrow C$ is in $\mathbb{C}'$. Hence, we can build the derivation $\mathcal{D}'$ obtained from $\mathcal{D}$ by replacing the ordered clause R labeling the root's child of $\mathcal{D}$ by $H \wedge \phi \wedge \phi_i \rightarrow C$. Since we only modified the formulas within the rule, we directly obtain that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ implies $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$. □

Lemma 44. *Consider the preamble* $\mathcal{P}_{\mathsf{verif}}$. *Let* $T \in \mathsf{trace}^{n_{IO}}_{IO}(\mathcal{C}_I, \rightarrow_i)$. *Let* $\varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \psi \wedge \bigwedge_{k=1}^{\ell} G_k^{\delta_k} \rightsquigarrow \psi_k)$.

Let $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ *be a tuple of steps. Let* $\tau_{G_1}, \ldots, \tau_{G_\ell}$ *be a tuple of steps such that for all* $k \in \{1, \ldots, \ell\}$, *for all* $i \in \mathrm{dom}(\delta_k)$, $\tau_{G_k}\ \delta_k(i)\ \tau_i$.

Let $\sigma, \sigma_s, \sigma_d$ *be three substitutions and* ϱ' *be the query* $\varrho' = (\bigwedge_{i=1}^{n} F_i\sigma_s \wedge \bigwedge_{k=1}^{\ell} G_k\sigma_s \Rightarrow \psi\sigma_s \wedge \bigwedge_{k=1}^{\ell} \psi_k^{\delta_k[n+k\mapsto\leq]}\sigma_s)$ *such that* $T, \tau_i \vdash^{n_{IO}}_{IO} F_i\sigma_s\sigma_d$ *for* $i = 1 \ldots n$ *and* $T, \tau_{G_k} \vdash^{n_{IO}}_{IO} G_k\sigma_s\sigma_d$ *for* $k = 1 \ldots \ell$.

By denoting $\tilde{\tau}' = (\tau_1, \ldots, \tau_n, \tau_{G_1}, \ldots, \tau_{G_\ell})$, *if there exists an annotated query conclusion* Ψ' *and a substitution* $\sigma_{\varrho'}$ *such that:*

- $\overline{\Psi'} = \psi\sigma_s \wedge \bigwedge_{k=1}^{\ell} \psi_k^{\delta_k[n+k\mapsto\leq]}\sigma_s$ *and for all* $F^{\delta,\mu} \in \Psi'$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}', \sigma_d)\}$
- $F_i\sigma = F_i\sigma_s\sigma_d = F_i\sigma_s\sigma_{\varrho'}$ *for* $i = 1 \ldots n$ *and* $G_k\sigma_s\sigma_d = G_k\sigma_s\sigma_{\varrho'}$ *for* $k = 1 \ldots \ell$
- $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models \Psi'\sigma_{\varrho'}$
- *for all* $F^{\delta,\mu} \in \Psi'$, *if* $\mu(\tilde{\tau}', \sigma_d)$ *is defined then* $T, \mu(\tilde{\tau}', \sigma_d) \vdash_i F\sigma_{\varrho'}$

then there exists an annotated query conclusion Ψ *and a substitution* σ_ϱ *such that:*

- $\overline{\Psi} = \psi \wedge \bigwedge_{k=1}^{\ell} G_k^{\delta_k} \rightsquigarrow \psi_k$ *and for all* $F^{\delta,\mu} \in \Psi$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$
- $F_i\sigma = F_i\sigma_\varrho$ *for* $i = 1 \ldots n$
- $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$
- *for all* $F^{\delta,\mu} \in \Psi$, *if* $\mu(\tilde{\tau}, \sigma)$ *is defined then* $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ *and either there exists* $F'^{\delta',\mu'} \in \Psi'$ *such that* $F'\sigma_{\varrho'} = F\sigma_\varrho$ *and* $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma_d)$ *or there exists* $k \in \{1, \ldots, \ell\}$ *such that* $G_k\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$.

Proof. Let us build Ψ from Ψ' such that $\overline{\Psi} = \psi \wedge \bigwedge_{k=1}^{\ell} G_k^{\delta_k} \rightsquigarrow \psi_k$ and for all $F^\delta \in \psi$ (resp. ψ_k for $k = 1 \ldots \ell$), if $F^\delta\sigma_s$ is annotated by μ' in Ψ' then in Ψ, we annotate F^δ by μ such that $\mathrm{dom}(\mu) = \emptyset$ iff $\mathrm{dom}(\mu') = \emptyset$ and $\mu'(\tilde{\tau}', \sigma_d)$ is defined implies that $\mu'(\tilde{\tau}', \sigma_d) = \mu(\tilde{\tau}, \sigma)$. Moreover, for all $k \in \{1, \ldots, \ell\}$, we annotate $G_k^{\delta_k}$ by μ_k such that $\mathrm{dom}(\mu_k) = \{(\tilde{\tau}, \sigma)\}$ and $\mu_k(\tilde{\tau}, \sigma) = \tau_{G_k}$. Let us define $\sigma_\varrho = \sigma_s\sigma_{\varrho'}$. Note that with $F_i\sigma = F_i\sigma_s\sigma_{\varrho'}$ for $i = 1 \ldots n$ by hypothesis, we trivially have that $F_i\sigma = F_i\sigma_\varrho$ for $i = 1 \ldots n$.

By hypothesis, we know that for all $k \in \{1, \ldots, \ell\}$, $G_k\sigma_s\sigma_d = G_k\sigma_s\sigma_{\varrho'} = G_k\sigma_\varrho$ and $T, \tau_{G_k} \vdash^{n_{IO}}_{IO} G_k\sigma_s\sigma_d$ and for all $i \in \mathrm{dom}(\delta_k)$, $\tau_{G_k}\ \delta_k(i)\ \tau_i$. With G_k being events, we obtain that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models G_k^{\delta_k,\mu_k}\sigma_\varrho$.

Similarly, for all $F^\delta \in \psi$, let us denote μ' and μ the partial functions associated to $F^\delta\sigma_s$ in Ψ' and to F^δ in Ψ respectively. We know that if $\mu'(\tilde{\tau}', \sigma_d)$ is defined then $\mu'(\tilde{\tau}', \sigma_d) = \mu(\tilde{\tau}, \sigma)$ and $T, \mu'(\tilde{\tau}', \sigma_d) \vdash_i F\sigma_s\sigma_{\varrho'}$. Hence $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$. Furthermore, $F^{\delta,\mu'}\sigma_s \in \Psi'$ and $\mu'(\tilde{\tau}', \sigma_d)$ defined implies that there exists $F'^{\delta'} \in \psi_\varrho$ such that $F'\sigma' = F\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$. Finally, since $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models \Psi'\sigma_{\varrho'}$, we also deduce that if $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models F^{\delta,\mu'}\sigma_s\sigma_{\varrho'}$ then for all $i \in \mathrm{dom}(\delta)$, $\mu'(\tilde{\tau}', \sigma_d)\ \delta(i)\ \tau_i$ and so $\mu(\tilde{\tau}, \sigma)\ \delta(i)\ \tau_i$. This allows us to deduce $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models F^{\delta,\mu}\sigma_\varrho$. This conclude the proof Item 4 of the lemma.

For all $k \in \{1, \ldots, \ell\}$, let us denote Ψ_k the restriction of Ψ to ψ_k, i.e. $\overline{\Psi_k} = \psi_k$. For all $F^\delta \in \psi_k$, let us denote μ' and μ the partial functions associated to $F^{\delta[n+k\mapsto\leq]}\sigma_s$ in Ψ' and to F^δ in Ψ respectively. Similarly to the previous case (i.e. $F^\delta \in \psi$), we have that if

$\mu'(\tilde{\tau}', \sigma_d)$ is defined then $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$. Moreover, $F^{\delta[n+k\mapsto\leq],\mu'}\sigma_s \in \Psi'$ and $\mu'(\tilde{\tau}', \sigma_d)$ defined implies that there exists $F'^{\delta'} \in \psi_\varrho$ such that $F'\sigma' = F\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$. Finally, since $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models \Psi'\sigma_{\varrho'}$, we also deduce that if $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models F^{\delta[n+k\mapsto\leq],\mu'}\sigma_s\sigma_{\varrho'}$ then for all $i \in \mathrm{dom}(\delta)$, $\mu'(\tilde{\tau}', \sigma_d)\ \delta(i)\ \tau_i$ and so $\mu(\tilde{\tau}, \sigma)\ \delta(i)\ \tau_i$; and $\mu'(\tilde{\tau}', \sigma_d) \leq \tau_{G_k}$. This allows us to prove that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_k\sigma_\varrho$ and for all $F^{\delta,\mu} \in \Psi_k$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $\mu(\tilde{\tau}, \sigma) \leq \tau_{G_k}$. Since we already proved that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models G_k^{\delta_k,\mu_k}\sigma_\varrho$ with $\mu_k(\tilde{\tau}, \sigma) = \tau_{G_k}$, we therefore obtain that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models G_k^{\delta_k,\mu_k}\sigma_\varrho \rightsquigarrow \Psi_k$.

This conclude the proof of $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$. □

E.2 Main proof

Lemma 45. *Consider the preamble $\mathcal{P}_{\mathsf{verif}}$. Assume that for all $\varrho = (\bigwedge_{i=1}^n F_i \Rightarrow \psi) \in \mathcal{Q} \cup \mathcal{L}_i$, there exists $\mathcal{S}_{ol\,\varrho}$ such that $\mathtt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ terminates and is true where $R_q = (G_1'^{[1\mapsto\leq]} \wedge \ldots \wedge G_n'^{[n\mapsto\leq]} \rightarrow \bigwedge_{i=1}^n G_i')$ and $G_i' = \lceil F_i \rceil^{may}$ for $i = 1 \ldots n$.*

For all $\varrho \in \mathcal{Q} \cup \mathcal{L}_i$, for all traces $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$, if $\varrho = (\bigwedge_{i=1}^n F_i \Rightarrow \psi)$ then there exists an annotated query conclusion Ψ such that

1. *for all $F^{\delta,\mu} \in \Psi$, for all $(\tau_1, \ldots, \tau_n, \sigma) \in \mathrm{dom}(\mu)$, $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$*
2. *for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ and all substitutions σ, if $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ then there exist $(R = (H \wedge \phi \rightarrow C), \sigma_s, \varrho_s) \in \mathcal{S}_{ol\,\varrho}$, a derivation $\mathcal{D}$ and two substitutions σ_ϱ, σ_d such that:*
   (a) *$\overline{\Psi} = \psi$, $F_i\sigma = F_i\sigma_\varrho$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$*
   (b) *ϱ_s is of the form $\bigwedge_{i=1}^m G_i \Rightarrow \psi_\varrho$ with $m \geq n$ and $F_i\sigma = G_i\sigma_s\sigma_d$ for $i \leq n$*
   (c) *$\mathcal{D}$ is a derivation of $\bigwedge_{i=1}^m \lceil G_i\sigma_s\sigma_d \rceil^{may}$ at steps $(\tau_{1,\varrho}, \ldots, \tau_{m,\varrho})$ from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and $\tau_i = \tau_{i,\varrho}$ for $i \leq n$, $C = \bigwedge_{i=1}^m \lceil G_i \rceil^{may}\sigma_s$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$*
   (d) *for all $F^{\delta,\mu} \in \Psi$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and either there exists $F'^{\delta'} \in \psi_\varrho$ such that $F'\sigma_s\sigma_d = F\sigma_\varrho$ or there exists $n + 1 \leq i \leq m$ such that $G_i\sigma_s\sigma_d = F\sigma_\varrho$.*

Proof. To prove this lemma, we need to prove an inside property.

Inside property: For all $n_{nest} \in \mathbb{N}$, for all $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$, for all $n \in \mathbb{N}$, for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, for all sets of queries $\mathcal{Q}'$, for all $\varrho \in \mathcal{Q}'$, for all substitutions σ, if

- $\mathcal{L}_i \subseteq \mathcal{Q}'$
- $\varrho = (\bigwedge_{i=1}^n F_i \Rightarrow \psi)$ and ϱ contains at most n_{nest} instances of the nested relation $\rightsquigarrow$
- there exist an ordered clause R_q, a set $\mathcal{S}_{ol\,\varrho}$ such that $\mathtt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ is true and
  - either $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ and $R_q = (G_1'^{[1\mapsto\leq]} \wedge \ldots \wedge G_n'^{[n\mapsto\leq]} \rightarrow \bigwedge_{i=1}^n G_i')$ and $G_i' = \lceil F_i \rceil^{may}$ for $i = 1 \ldots n$
  - or there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^n \lceil F_i \rceil^{may}\sigma$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$,

then there exists an annotated query conclusion Ψ such that:

1. for all $F^{\delta,\mu} \in \Psi$, $\text{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$
2. there exist $(R = (H \wedge \phi \rightarrow C), \sigma_s, \varrho_s) \in \mathcal{S}_{ol\,\varrho}$, a derivation $\mathcal{D}$ and two substitutions σ_ϱ, σ_d such that:
   (a) $\overline{\Psi} = \psi$, $F_i\sigma = F_i\sigma_\varrho$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$
   (b) ϱ_s is of the form $\bigwedge_{i=1}^{m} G_i \Rightarrow \psi_\varrho$ with $m \geq n$ and $F_i\sigma = G_i\sigma_s\sigma_d$ for $i \leq n$
   (c) $\mathcal{D}$ is a derivation of $\bigwedge\!\!\!\!\bigwedge_{i=1}^{m} \lceil G_i\sigma_s\sigma_d \rceil^{may}$ at steps $(\tau_{1,\varrho}, \ldots, \tau_{m,\varrho})$ from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and $\tau_i = \tau_{i,\varrho}$ for $i \leq n$, $C = \bigwedge\!\!\!\!\bigwedge_{i=1}^{m} \lceil G_i\sigma_s \rceil^{may}$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$
   (d) for all $F^{\delta,\mu} \in \Psi$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and either there exists $F'^{\delta'} \in \psi_\varrho$ such that $F'\sigma_s\sigma_d = F\sigma_\varrho$ or there exists $n + 1 \leq i \leq m$ such that $G_i\sigma_s\sigma_d = F\sigma_\varrho$.

Proof of the main result: Assume for now that the inside property is true. We show how we can derive from it the main result. Notice that the main difference between the main result and the inside property is the inversion of quantifier between the existential quantification of annotated query conclusion Ψ and the universal quantification of the tuples of steps $\tilde{\tau}$ and substitution σ. More specifically, since $\texttt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ terminates and is true where $R_q = (G_1'^{[1\mapsto\leq]} \wedge \ldots \wedge G_n'^{[n\mapsto\leq]} \rightarrow \bigwedge\!\!\!\!\bigwedge_{i=1}^{n} G_i')$ and $G_i' = \lceil F_i \rceil^{may}$ for $i = 1 \ldots n$, by the inside property, we know that for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ and all substitutions σ, if $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ then there exists an annotated query conclusion $\Psi_{\tilde{\tau},\sigma}$ satisfying Items 1 and 2. However, Item 1 of the inside property indicates that the annotated query conclusion $\Psi_{\tilde{\tau},\sigma}$ is built such that for all $F^{\delta,\mu} \in \Psi$, $\text{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$, that is, either μ is not defined or μ is only defined on $(\tilde{\tau}, \sigma)$.

Hence, we can build an annotated query conclusion Ψ such that for all instances of F^δ in ψ, we associate the partial function μ' where for all $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ and all substitutions σ,

- if $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ then $\mu'_{|(\tilde{\tau},\sigma)} = \mu$ where μ is the partial function associated to this instance of F^δ in $\Psi_{\tilde{\tau},\sigma}$
- $\mu'(\tilde{\tau}, \sigma)$ is not defined otherwise.

Notice that Ψ is well defined since the partial functions $\Psi_{\tilde{\tau},\sigma}$ and $\Psi_{\tilde{\tau}',\sigma'}$ necessarily have disjoint domain when $(\tilde{\tau}, \sigma) \neq (\tilde{\tau}', \sigma')$.

By construction, we directly have that Ψ satisfy Item 1 of the main result. Furthermore, since for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ and all substitutions σ, if $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ then the annotated query conclusion $\Psi_{\tilde{\tau},\sigma}$ satisfies Item 2 of the inside property, we also deduce that Ψ satisfies Item 2 of the main result which allows us to conclude.

Proof of inside property: We prove this property by induction on $(n_{nest}, T, \tilde{\tau})$ with the order $<$ defined as $(n_{nest}, T, \tilde{\tau}) < (n'_{nest}, T', \tilde{\tau}')$ when $n_{nest} < n'_{nest}$ or $(n_{nest} = n'_{nest}$ and $(T, \tilde{\tau}) <_{ind} (T', \tilde{\tau}'))$.

In the base case, $n_{nest} = 0$, T is the empty trace and $\tilde{\tau}$ is the empty tuple. Since there is no query of the form $\bigwedge_{i=1}^{n} F_i \Rightarrow \psi$ with $n = 0$ then the result trivially holds.

In the inductive step, let $\mathcal{Q}'$ be a set of queries, $\varrho = (\bigwedge_{i=1}^{n} F_i \Rightarrow \psi) \in \mathcal{Q}'$ and σ a substitution such that:

- $\mathcal{L}_i \subseteq \mathcal{Q}'$
- ϱ contains at most n_{nest} instances of the nested relation $\rightsquigarrow$
- there exist an ordered clause R_q, a set $\mathcal{S}_{ol\,\varrho}$ such that $\texttt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ is true and
  - either $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \dots n$ and $R_q = (G_1'^{[1 \mapsto \leq]} \wedge \dots \wedge G_n'^{[n \mapsto \leq]} \rightarrow \bigwedge_{i=1}^{n} G_i')$ and $G_i' = \lceil F_i \rceil^{may}$ for $i = 1 \dots n$
  - or R_q satisfies $\mathcal{S}_p$ and there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$,

Note if ϱ contains $n'_{next} < n_{nest}$ instances of the nested relation $\rightsquigarrow$, we can directly conclude by applying our inductive hypothesis on $(n'_{nest}, T, \tilde{\tau})$. Thus, let us assume that ϱ contains n_{nest} instances of the nested relation $\rightsquigarrow$.

We show that when $n_{nest} = 0$, $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ holds and when $n_{nest} > 0$, $\mathcal{H}yp_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(T, \tilde{\tau})$ holds. By our preamble $\mathcal{P}_{\mathsf{verif}}$, we know that for all $\varrho' \in \mathcal{L}$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models \varrho'$. Let $\varrho' = (\bigwedge_{i=1}^{n'} F_i' \Rightarrow \psi') \in \mathcal{L}_i$. By hypothesis of the lemma, we know that there exists $\mathcal{S}_{ol\,\varrho'}$ such that $\texttt{verify}(\varrho', R_q', \mathcal{S}_{ol\,\varrho'})$ terminates and is true where $R_q' = (G_1'^{[1 \mapsto \leq]} \wedge \dots \wedge G_{n'}'^{[n \mapsto \leq]} \rightarrow \bigwedge_{i=1}^{n'} G_i')$ and $G_i' = \lceil F_i' \rceil^{may}$ for $i = 1 \dots n'$.

- Case $n_{nest} = 0$: Let $T' \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$ and let $\tilde{\tau}' = (\tau_1', \dots, \tau_{n'}')$ be a tuple of steps such that $(T', \tilde{\tau}') <_{ind} (T, \tilde{\tau})$. By definition, ϱ' is non-nested and non-injective. To show that $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ holds, we need to prove that $\mathcal{IH}_{T',\tilde{\tau}'}$ holds. Let σ' be a substitution. If $T', \tau_i' \vdash_{IO}^{n_{IO}} F_i'\sigma'$ for $i = 1 \dots n'$ then we can apply our inductive hypothesis on $(0, T', \tilde{\tau}')$ to obtain that there exists an annotated query conclusion $\Psi_{\sigma'}$ and a substitution $\sigma_{\varrho'}$ such that
  - for all $F^{\delta,\mu} \in \Psi_{\sigma'}$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}', \sigma')\}$
  - $\overline{\Psi_{\sigma'}} = \psi'$, $F_i'\sigma' = F_i'\sigma_{\varrho'}$ for $i = 1 \dots n'$ and $(\vdash_i, T', (\tilde{\tau}', \sigma')) \models \Psi_{\sigma'}\sigma_{\varrho'}$.

  Hence similarly to the proof of the main result, since the domain of partial functions are disjoint from two annotated query conclusions $\Psi_{\sigma_1'}$ and $\Psi_{\sigma_2'}$ with $\sigma_1' \neq \sigma_2'$, we can create an annotated query conclusion Ψ such that for all instances of F^δ in ψ', we associate the partial function μ' where for all substitutions σ',
  - if $T', \tau_i' \vdash_{IO}^{n_{IO}} F_i'\sigma'$ for $i = 1 \dots n'$ then $\mu'_{|(\tilde{\tau}', \sigma')} = \mu$ where μ is the partial function associated to this instance of F^δ in $\Psi_{\sigma'}$
  - $\mu'(\tilde{\tau}', \sigma')$ is not defined otherwise.

  Once again, Ψ is well defined and satisfies for all substitutions σ', if $T', \tau_i' \vdash_{IO}^{n_{IO}} F_i'\sigma'$ for $i = 1 \dots n'$ then there exists $\sigma_{\varrho'}$ such that $F_i'\sigma' = F_i'\sigma_{\varrho'}$ for $i = 1 \dots n'$ and $(\vdash_i, T', (\tilde{\tau}', \sigma')) \models \Psi\sigma_{\varrho'}$ which allows us to deduce that $\mathcal{IH}_{T',\tilde{\tau}'}$ holds and so $\mathcal{H}yp_{\mathcal{L},\mathcal{L}_i}(T, \tilde{\tau})$ holds.

- Case $n_{nest} > 0$: To prove $\mathcal{H}yp_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(T, \tilde{\tau})$, we need to show that for all $\varrho' \in \mathcal{L}_i$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models \varrho'$. The proof is in fact very similar to the case $n_{nest} = 0$. Let $T' \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$, let $\tilde{\tau}' = (\tau_1', \dots, \tau_{n'}')$ be a tuple of steps and σ' a substitution such that $T, \tau_i' \vdash_{IO}^{n_{IO}} F_i'\sigma'$ for $i = 1 \dots n'$. Since ϱ' is non-nested and non-injective, we can apply our inductive hypothesis on $(0, T', \tilde{\tau}')$ to obtain that there exists an annotated query conclusion $\Psi_{\tilde{\tau}',\sigma'}$ and a substitution $\sigma_{\varrho'}$ such that

– for all $F^{\delta,\mu} \in \Psi_{\tilde{\tau}',\sigma'}$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}', \sigma')\}$
– $\overline{\Psi_{\tilde{\tau}',\sigma'}} = \psi'$, $F'_i\sigma' = F'_i\sigma_{\varrho'}$ for $i = 1 \dots n'$ and $(\vdash_i, T', (\tilde{\tau}', \sigma')) \models \Psi_{\tilde{\tau}',\sigma'}\sigma_{\varrho'}$.

Once again, since the domain of partial functions are disjoint from two annotated query conclusions $\Psi_{\tilde{\tau}'_1,\sigma'_1}$ and $\Psi_{\tilde{\tau}'_2,\sigma'_2}$ with $(\tilde{\tau}'_1, \sigma'_1) \neq (\tilde{\tau}'_2, \sigma'_2)$, we can create an annotated query conclusion Ψ such that for all instances of F^δ in ψ', we associate the partial function μ' where for all tuples of steps $\tilde{\tau}' = (\tau'_1, \dots, \tau'_{n'})$, for all substitutions σ',

– if $T', \tau'_i \vdash_{IO}^{n_{IO}} F'_i\sigma'$ for $i = 1 \dots n'$ then $\mu'_{|(\tilde{\tau}',\sigma')} = \mu$ where μ is the partial function associated to this instance of F^δ in $\Psi_{\tilde{\tau}',\sigma'}$
– $\mu'(\tilde{\tau}', \sigma')$ is not defined otherwise.

Ψ is well defined and satisfies for all tuples of steps $\tilde{\tau}' = (\tau'_1, \dots, \tau'_{n'})$, for all substitutions σ', if $T', \tau'_i \vdash_{IO}^{n_{IO}} F'_i\sigma'$ for $i = 1 \dots n'$ then there exists $\sigma_{\varrho'}$ such that $F'_i\sigma' = F'_i\sigma_{\varrho'}$ for $i = 1 \dots n'$ and $(\vdash_i, T', (\tilde{\tau}', \sigma')) \models \Psi\sigma_{\varrho'}$. This allows us to conclude that $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models \varrho'$.

Now that we proved the two properties on the inductive lemmas, we can complete the proof by doing another case analysis on n_{nest}. Recall that by construction, $\mathbb{C}_{sat}$ is a set of selection free, simplified clauses. Moreover, from Lemmas 28 and 31, we also know that $\mathbb{C}_{sat}$ is a set of well-originated clauses containing the selection free clauses of $\mathbb{C}_{std}$.

- Case $n_{nest} = 0$: In such a case, we know that ψ is of the form $\bigvee_{j=1}^{m} \psi_j$. Moreover, since $\mathtt{verify}(\varrho, R_q, \mathcal{S}_{ol_\varrho})$ is true and by denoting $\mathbb{C}_s = \mathsf{saturate}\mathsf{S}_{\mathcal{L},\mathcal{L}_i}^{\mathcal{S}_p}(\{R_q\}, \mathbb{C}_{sat})$, we deduce that there exists $\mathbb{C}'_s$ completely covering $\mathbb{C}_s$ such that for all $R \in \mathbb{C}'_s$, there exists $j_0 \in \{1, \dots, m\}$, a substitution σ_s and a matching $\mathbb{M}$ such that $R, \sigma_s, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0}$ and $(R, \sigma, \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0}) \in \mathcal{S}_{ol_\varrho}$.

  By hypothesis, we know that either (i) there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$, or (ii) $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \dots n$ and $R_q = (G_1'^{[1 \mapsto \leq]} \wedge \dots \wedge G_n'^{[n \mapsto \leq]} \rightarrow \bigwedge_{i=1}^{n} G'_i)$ and $G'_i = \lceil F_i \rceil^{may}$ for $i = 1 \dots n$. In Case (ii), by applying Lemma 40 we obtain that Case (i) is also satisfied. Hence we deduce that there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$.

  Note that the ordering functions in R_q do not have $<$ in their image. Hence R_q directly satisfies $\mathcal{S}_p$. We can therefore apply Theorem 4 and lemma 43 to obtain that there exists a derivation $\mathcal{D}$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\mathbb{C}'_s$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$.

  Let $R = (H \wedge \phi \rightarrow C)$ be the ordered clause of $\mathbb{C}'_s$ labeling the root's child of $\mathcal{D}$. We know that $R, \sigma_s, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0}$. Hence, by Lemma 42, we deduce that there exists a annotated query conclusion Ψ_{j_0} and a substitution σ_d such that:

  – $C = \bigwedge_{i=1}^{n} \lceil F_i\sigma_s \rceil^{may}$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$;
  – $F_i\sigma_s\sigma_d = F_i\sigma$ for $i = 1 \dots n$ and $\overline{\Psi_{j_0}} = \psi_{j_0}$;
  – $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_s\sigma_d$;
  – for all $F^{\delta,\mu} \in \Psi_{j_0}$, $\mathrm{dom}(\mu) = \{(\tilde{\tau}, \sigma)\}$

Let us build Ψ by extending Ψ_{j_0} to ψ. More specifically, $\Psi = \bigvee_{j=1}^{m} \Psi'_j$ where $\Psi'_{j_0} = \Psi_{j_0}$ and for all $j \neq j_0$, $\overline{\Psi'_j} = \psi_j$ and for all $F^{\delta,\mu} \in \Psi'_j$, $\mathrm{dom}(\mu) = \emptyset$. Intuitively, all partial functions μ are undefined for all $j \neq j_0$.

Since for all $F^{\delta,\mu} \in \Psi_{j_0}$, $\mathrm{dom}(\mu) = \{(\tilde{\tau}, \sigma)\}$, then by construction of Ψ, we deduce that for all $F^{\delta,\mu} \in \Psi$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$ which allows us to prove Item 1 of the inside property.

Let us define $\sigma_\varrho = \sigma_s\sigma_d$. Since $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_s\sigma_d$, we have $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_\varrho$. Since $\Psi = \bigvee_{j=1}^{m} \Psi'_j$ with $\Psi'_{j_0} = \Psi_{j_0}$, we deduce that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$. Moreover, since $F_i\sigma_s\sigma_d = F_i\sigma$ for $i = 1 \dots n$, we obtain $F_i\sigma_\varrho = F_i\sigma$ for $i = 1 \dots n$ and so Item 2a of the inside property holds.

We already showed that $(R, \sigma, \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0}) \in \mathcal{S}_{ol\,\varrho}$, hence by taking $m = n$, $G_i = F_i$ for $i = 1 \dots n$ and $\psi_\varrho = \psi_{j_0}$, we directly obtain Items 2b and 2c of the inside property.

Finally, by construction of Ψ, for all $F^{\delta,\mu} \in \Psi$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $F^{\delta,\mu} \in \Psi_{j_0}$ and $F^{\delta} \in \psi_{j_0}$. Since Ψ_{j_0} is a conjunction and so does not contain any disjunction, $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_s\sigma_d$ implies that $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_s\sigma_d$. With $\sigma_\varrho = \sigma_s\sigma_d$, we conclude that $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and there exists $F'^{\delta'} \in \psi_{j_0}$ such that $F'\sigma_s\sigma_d = F\sigma_\varrho$ (by taking $F' = F$ and $\delta' = \delta$). This allows us to prove Item 2d of the inside property.

- Case $n_{nest} > 0$: In such a case, we know that ψ is of the form $\bigvee_{j=1}^{m}(\psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j})$. Moreover, since $\texttt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ terminates, is true and by denoting $\mathbb{C}_s = \mathsf{saturate}\mathsf{S}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(\{R_q\}, \mathbb{C}_{sat})$, we deduce that there exists $\mathbb{C}'_s$ completely covering $\mathbb{C}_s$ such that for all $R \in \mathbb{C}'_s$, there exists $j_0 \in \{1, \dots, m\}$, a substitution σ_s and a matching $\mathbb{M}$ such that

  - $R, \sigma_s, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}}$
  - $\texttt{verify}(\varrho', R'_q, \mathcal{S}_{ol\,\varrho})$
  - $(R'_q, \varrho') = \texttt{gen\_nested}(\bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}} \rightsquigarrow \psi_{k,j_0}, (R, \sigma_s, \mathbb{M}))$

  Finally, by our hypotheses, we know that either (i) there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$, or (ii) $T, \tau_i \vdash^{n_{IO}}_{IO} F_i\sigma$ for $i = 1 \dots n$ and $R_q = (G_1'^{[1\mapsto\leq]} \wedge \dots \wedge G_n'^{[n\mapsto\leq]} \rightarrow \bigwedge_{i=1}^{n} G'_i)$ and $G'_i = \lceil F_i \rceil^{may}$ for $i = 1 \dots n$. We already proved that $\mathcal{H}yp_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(T, \tilde{\tau})$. Hence in Case (ii), by applying Lemma 40 we obtain that Case (i) is also satisfied. Hence we deduce that there exists a derivation $\mathcal{D}_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\{R_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_q$.

  We can now apply Theorem 4 and lemma 43 to obtain that there exists a derivation $\mathcal{D}'_q$ of $\bigwedge_{i=1}^{n} \lceil F_i\sigma \rceil^{may}$ at steps $\tilde{\tau}$ from $\mathbb{C}'_s$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'_q$.

  Let $R = (H \wedge \phi \rightarrow C)$ be the ordered clause of $\mathbb{C}'_s$ labeling the root's child of $\mathcal{D}'_q$. We know that that there exists $j_0 \in \{1, \dots, m\}$ such that $R, \sigma_s, \mathbb{M} \models \bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}}$. Hence, by Lemma 42, we deduce that there exists a annotated query conclusion $\Psi^e_{j_0}$ and a substitution σ_d such that:

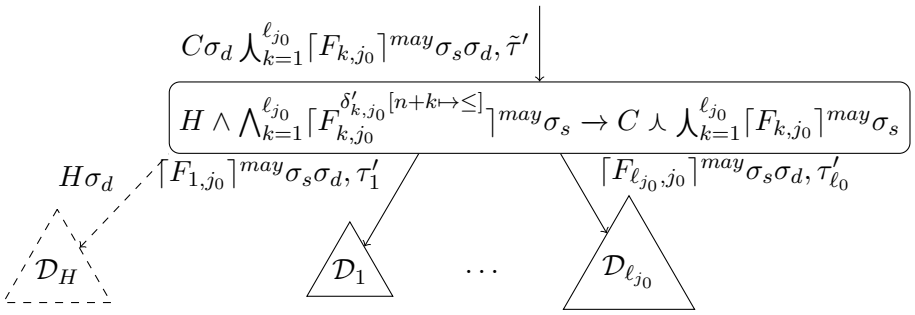
  - $C = \bigwedge_{i=1}^{n} \lceil F_i\sigma_s \rceil^{may}$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$;

– $F_i\sigma_s\sigma_d = F_i\sigma$ for $i = 1 \dots n$ and $\overline{\Psi^e_{j_0}} = \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}}$;
– $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi^e_{j_0}\sigma_s\sigma_d$
– for all $k \in \{1, \dots, \ell_{j_0}\}$, if we denote by μ_k the partial function associated to $F_{k,j_0}^{\delta_{k,j_0}}$ in $\Psi^e_{j_0}$ then there exists δ'_{k,j_0} such that $(F_{k,j_0}^{\delta_{k,j_0}}, \delta'_{k,j_0}) \in \mathbb{M}$ and for all $i \in \mathrm{dom}(\delta'_{k,j_0})$, $\mu_k(\tilde{\tau}, \sigma)\ \delta'_{k,j_0}(i)\ \tau_i$.

Note that since $\psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}}$ is only a conjunction of facts and formulas, we deduce from $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi^e_{j_0}\sigma_s\sigma_d$ that for all $k \in \{1, \dots, \ell_{j_0}\}$, $T, \mu_k(\tilde{\tau}, \sigma) \vdash_i F_{k,j_0}\sigma_s\sigma_d$. Note that F_{k,j_0} being an event, we have $T, \mu_k(\tilde{\tau}, \sigma) \vdash_{IO}^{n_{IO}} F_{k,j_0}\sigma_s\sigma_d$ and so $T, \mu_k(\tilde{\tau}, \sigma) \vdash_{IO}^{n_{IO}} \lceil F_{k,j_0}\sigma_s\sigma_d \rceil^{may}$. Since we already proved that $\mathcal{H}yp_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(T, \tilde{\tau})$ holds, we deduce from Theorems 1 and 2 that there exists a derivation $\mathcal{D}_k$ of $\lceil F_{k,j_0}\sigma_s\sigma_d \rceil^{may}$ at step $\mu_k(\tilde{\tau}, \sigma)$ from $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_k$.

Consider the annotated clause $R'_q = H \wedge \bigwedge_{k=1}^{\ell_{j_0}} \lceil F_{k,j_0}^{\delta'_{k,j_0}[n+k\mapsto\leq]} \rceil^{may}\sigma_s \to C \curlywedge \curlywedge_{k=1}^{\ell_{j_0}} \lceil F_{k,j_0} \rceil^{may}\sigma_s$. If we denote $\tau'_1, \dots, \tau'_{\ell_{j_0}}$ the steps $\mu_1(\tilde{\tau}, \sigma), \dots, \mu_{\ell_{j_0}}(\tilde{\tau}, \sigma)$ respectively and we denote $\tilde{\tau}' = \tau_1, \dots, \tau_n, \tau'_1, \dots, \tau'_{\ell_{j_0}}$ then we can build the ordered derivation $\mathcal{D}_q$ of $C\sigma_d \curlywedge \curlywedge_{k=1}^{\ell_{j_0}} \lceil F_{k,j_0} \rceil^{may}\sigma_s\sigma_d$ at step $\tilde{\tau}'$ from $\{R'_q\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ as follows:

The derivation $\mathcal{D}_q$



Consider now the query $\varrho' = \bigwedge_{i=1}^{n} F_i\sigma_s \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}\sigma_s \Rightarrow \psi_{j_0}\sigma_s \wedge \bigwedge_{k=1}^{\ell_{j_0}} \psi_{k,j_0}^{\delta_{k,j_0}[n+k\mapsto\leq]}\sigma_s$. Since R'_q, ϱ' are in fact the result of $\texttt{gen\_nested}(\bigwedge_{i=1}^{n} F_i \Rightarrow \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}}, (R, \sigma_s, \mathbb{M}))$; $\texttt{verify}(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ implies $\texttt{verify}(\varrho', R'_q, \mathcal{S}_{ol\,\varrho})$ and ϱ' contains strictly less nested queries than ϱ, we can apply our inductive hypothesis on it with the trace T, the tuple of steps $\tilde{\tau}'$, the set of query $\{\varrho'\} \cup \mathcal{L}_i$ and the substitution σ_d.

This allows us to deduce that there exists an annotated query conclusion Ψ'_{j_0} such that:

1. for all $F^{\delta,\mu} \in \Psi'_{j_0}$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}', \sigma_d)\}$
2. there exists $(R' = (H' \wedge \phi' \to C'), \sigma'_s, \varrho_s) \in \mathcal{S}_{ol\,\varrho'}$, a derivation $\mathcal{D}$ and two substitutions $\sigma_{\varrho'}$,σ'_d such that:
   (a) $\overline{\Psi'_{j_0}} = \psi_{j_0}\sigma_s \wedge \bigwedge_{k=1}^{\ell_{j_0}} \psi_{k,j_0}^{\delta_{k,j_0}[n+k\mapsto\leq]}\sigma_s$, $F_i\sigma_s\sigma_d = F_i\sigma_s\sigma_{\varrho'}$ for $i = 1 \dots n$ and $F_{k,j_0}\sigma_s\sigma_d = F_{k,j_0}\sigma_s\sigma_{\varrho'}$ for $k = 1 \dots \ell_{j_0}$ and $(\vdash_i, T, (\tilde{\tau}', \sigma_d)) \models \Psi'_{j_0}\sigma_{\varrho'}$
   (b) ϱ_s is of the form $\bigwedge_{i=1}^{m'} G_i \Rightarrow \psi_{\varrho'}$ with $m' \geq n + \ell_{j_0}$ and $F_i\sigma_s\sigma_d = G_i\sigma'_s\sigma'_d$ for $i \leq n$ and $F_{k,j_0}\sigma_s\sigma_d = G_{n+k}\sigma'_s\sigma'_d$ for $1 \leq k \leq \ell_{j_0}$.

(c) $\mathcal{D}$ is a derivation of $\bigwedge_{i=1}^{m'} \lceil G_i\sigma'_s\sigma'_d \rceil^{may}$ at steps $(\tau_{1,\varrho'}, \ldots, \tau_{m',\varrho'})$ from $\{R'\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and $\tau_i = \tau_{i,\varrho'}$ for $i \leq n$ and $\tau'_k = \tau_{n+k,\varrho'}$ for $1 \leq k \leq \ell_{j_0}$, $C' = \bigwedge_{i=1}^{m'} \lceil G_i\sigma'_s \rceil^{may}$, $\sigma'_d \vdash_i \phi'$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H'\sigma'_d$.

(d) for all $F^{\delta,\mu} \in \Psi'_{j_0}$, if $\mu(\tilde{\tau}', \sigma_d)$ is defined then $T, \mu(\tilde{\tau}', \sigma_d) \vdash_i F\sigma_{\varrho'}$ and either there exists $F'^{\delta'} \in \psi_{\varrho'}$ such that $F'\sigma'_s\sigma'_d = F\sigma_{\varrho'}$ or there exists $n + \ell_{j_0} + 1 \leq i \leq m'$ such that $G_i\sigma'_s\sigma'_d = F\sigma_{\varrho'}$.

Thanks to Items 1, 2.a and 2.d, we can apply Lemma 44 to obtain that there exists an annotated query conclusion Ψ_{j_0} and a substitution σ_ϱ such that:

- $\overline{\Psi_{j_0}} = \psi_{j_0} \wedge \bigwedge_{k=1}^{\ell_{j_0}} F_{k,j_0}^{\delta_{k,j_0}} \rightsquigarrow \psi_{k,j_0}$ and for all $F^{\delta,\mu} \in \Psi_{j_0}$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$
- $F_i\sigma = F_i\sigma_\varrho$ for $i = 1 \ldots n$
- $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_\varrho$
- for all $F^{\delta,\mu} \in \Psi_{j_0}$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and either there exists $F'^{\delta',\mu'} \in \Psi'_{j_0}$ such that $F'\sigma_{\varrho'} = F\sigma_\varrho$ and $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma_d)$ or there exists $k \in \{1, \ldots, \ell_{j_0}\}$ such that $F_{k,j_0}\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$.

Since $\varrho = \bigwedge_{i=1}^{n} F_i \Rightarrow \bigvee_{j=1}^{m}(\psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j})$, we can build an annotated query conclusion Ψ by extending Ψ_{j_0} to ψ. Moreover, specifically, $\Psi = \bigvee_{j=1}^{m} \Psi''_j$ where $\Psi''_{j_0} = \Psi_{j_0}$ and for all $j \neq j_0$, $\overline{\Psi''_j} = \psi_j \wedge \bigwedge_{k=1}^{\ell_j} F_{k,j}^{\delta_{k,j}} \rightsquigarrow \psi_{k,j}$ and for all $F^{\delta,\mu} \in \Psi''_j$, $\mathrm{dom}(\mu) = \emptyset$. Note that by construction of Ψ, we deduce that for all $F^{\delta,\mu} \in \Psi$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$.

Moreover, since $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi_{j_0}\sigma_\varrho$ and Ψ is a disjunction with Ψ_{j_0} as one of its disjuncts, we obtain that $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$.

Finally, let $F^{\delta,\mu} \in \Psi$ such that $\mu(\tilde{\tau}, \sigma)$ is defined. By construction of Ψ, we know that $F^{\delta,\mu} \in \Psi_{j_0}$. Hence, $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and either (i) there exists $F'^{\delta',\mu'} \in \Psi'_{j_0}$ such that $F'\sigma_{\varrho'} = F\sigma_\varrho$ and $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma_d)$, or (ii) there exists $k \in \{1, \ldots, \ell_{j_0}\}$ such that $F_{k,j_0}\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$ and $\mu(\tilde{\tau}, \sigma) = \tau'_k$. In Case (i), since $F'^{\delta',\mu'} \in \Psi'_{j_0}$, we know from Item 2.d that either there exists $F''^{\delta''} \in \psi_{\varrho'}$ such that $F''\sigma'_s\sigma'_d = F'\sigma_{\varrho'} = F\sigma_\varrho$ or there exists $n+\ell_{j_0}+1 \leq i \leq m'$ and so $n+1 \leq i \leq m'$ such that $G_i\sigma'_s\sigma'_d = F'\sigma_{\varrho'} = F\sigma_\varrho$. In Case (ii), we know from Item 2.b that $F_{k,j_0}\sigma_s\sigma_d = G_{n+k}\sigma'_s\sigma'_d$. Moreover, by Item 2.a, we know that $F_{k,j_0}\sigma_s\sigma_d = F_{k,j_0}\sigma_s\sigma_{\varrho'} = F\sigma_\varrho$. Hence, we deduce that $F\sigma_\varrho = G_{n+k}\sigma'_s\sigma'_d = G_i\sigma'_s\sigma'_d$ with $n + 1 \leq i \leq m'$.

To summarize, we have that an annotated query conclusion Ψ, $(R' = (H' \wedge \phi' \rightarrow C'), \sigma'_s, \varrho_s) \in \mathcal{S}_{ol\,\varrho}$, a derivation $\mathcal{D}$ and two substitutions $\sigma_\varrho, \sigma'_d$ such that:

- for all $F^{\delta,\mu} \in \Psi$, $\mathrm{dom}(\mu) \subseteq \{(\tilde{\tau}, \sigma)\}$
- $\overline{\Psi} = \psi$, $F_i\sigma = F_i\sigma_\varrho$ for $i = \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$
- ϱ_s is of the form $\bigwedge_{i=1}^{m'} G_i \Rightarrow \psi_{\varrho'}$ with $m' \geq n$ and $F_i\sigma = G_i\sigma'_s\sigma'_d$ for $i \leq n$
- $\mathcal{D}$ is a derivation of $\bigwedge_{i=1}^{m'} \lceil G_i\sigma'_s\sigma'_d \rceil^{may}$ at steps $(\tau_{1,\varrho'}, \ldots, \tau_{m',\varrho'})$ from $\{R'\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and $\tau_i = \tau_{i,\varrho'}$ for $i \leq n$, $C' = \bigwedge_{i=1}^{m'} \lceil G_i\sigma'_s \rceil^{may}$, $\sigma'_d \vdash_i \phi'$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H'\sigma'_d$.

– for all $F^{\delta,\mu} \in \Psi$, if $\mu(\tilde{\tau}, \sigma)$ is defined then $T, \mu(\tilde{\tau}, \sigma) \vdash_i F\sigma_\varrho$ and either there exists $F''^{\delta''} \in \psi_{\varrho'}$ such that $F''\sigma'\sigma'_d = F\sigma_\varrho$ or there exists $n+1 \leq i \leq m'$ such that $G_i\sigma'_s\sigma'_d = F\sigma_\varrho$.

This allows us to conclude our proof. □

Corollary 10. *Consider the preamble $\mathcal{P}_{\mathsf{verif}}$. Assume that for all $\varrho = (\bigwedge_{i=1}^n F_i \Rightarrow \psi) \in \mathcal{Q} \cup \mathcal{L}_i$, there exists $\mathcal{S}_{ol\,\varrho}$ such that* `verify`$(\varrho, R_q, \mathcal{S}_{ol\,\varrho})$ *terminates and is true where $R_q = (G_1'^{[1\mapsto\leq]} \wedge \ldots \wedge G_n'^{[n\mapsto\leq]} \rightarrow \bigwedge_{i=1}^n G'_i)$ and $G'_i = \lceil F_i \rceil^{may}$ for $i = 1 \ldots n$.*
For all $\varrho \in \mathcal{L}_i$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models \varrho$.

Proof. Direct from Lemma 45 and more specifically Item 2a. □

Theorem 6. *Let $\mathcal{C} = \mathcal{E}, P, \mathcal{A}$ be an initial configuration and $\mathcal{C}_I$ be is associated initial instrumented configuration. Let $\mathcal{L}$ be a sets of lemmas. Let $\mathcal{R}$ be a set of restrictions. Let $\mathcal{Q}$ be a set of correspondence queries.*
If the following holds:

- *for all $\varrho \in \mathcal{L} \cup \mathcal{Q} \cup \mathcal{R}$, $names(\varrho) \subseteq \mathcal{E}$*
- *for all $\varrho \in \mathcal{L}$, $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o))_{|\mathcal{R}} \models \varrho$*
- `prove`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{R}]_i, [\mathcal{Q}]_i)$ *terminates and returns true*

then for all $\varrho \in \mathcal{Q}$, $(\vdash_o, \mathsf{trace}(\mathcal{C}, \rightarrow_o)_{|\mathcal{R}}) \models \varrho$.

Proof. Consider $\mathcal{C}_I$ the initial instrumented configuration associated to $\mathcal{C}$. Since for all $\varrho \in \mathcal{Q} \cup \mathcal{L}$, $names(\varrho) \subseteq \mathcal{E}$, we can apply Lemma 3 meaning that for all $\varrho \in \mathcal{L}$, $(\vdash_i, \mathsf{trace}_i(\mathcal{C}_I, \rightarrow_i)) \models [\varrho]_i$ and it suffices to prove that for all $\varrho \in \mathcal{Q}$, $(\vdash_i, \mathsf{trace}_i(\mathcal{C}_I, \rightarrow_i)) \models [\varrho]_i$.

Note that by construction, in `prove`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{Q}]_i)$, we take n_{IO} such that all queries in $[\mathcal{Q}]_i$ are IO-n_{IO}-compliant. Hence, applying Lemma 8, we deduce that for all $\varrho \in \mathcal{L}$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models [\varrho]_i$. and it suffices to prove that for all $\varrho \in \mathcal{Q}$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models [\varrho]_i$.

Let us assume $[\mathcal{Q}]_i = \{\varrho_1, \ldots, \varrho_k\}$. Since `prove`$(\mathcal{C}_I, [\mathcal{L}]_i, [\mathcal{Q}]_i)$ terminates and is true, we can apply Lemma 45 hence we deduce the existence of the sets of solutions $\mathcal{S}_{ol1}, \ldots, \mathcal{S}_{olk}$ satisfying the properties stated in Lemma 45.

Let us focus on one query of $[\mathcal{Q}]_i$: Let $j \in \{1, \ldots, k\}$. Consider $\varrho_j = (\bigwedge_{i=1}^n F_i \Rightarrow \psi)$ and let $T \in \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)$. Thanks to Lemma 45, we know that there exists an annotated query conclusion Ψ satisfying Items 1 and 2 of Lemma 45.

In particular, Item 2a gives us that for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$ and all substitutions σ, if $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ for $i = 1 \ldots n$ then there exists a substitution σ_ϱ such that $\overline{\Psi} = \psi$, $F_i\sigma = F_i\sigma_\varrho$ for $i = 1 \ldots n$ and $(\vdash_i, T, (\tilde{\tau}, \sigma)) \models \Psi\sigma_\varrho$. Hence, to prove $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \rightarrow_i)) \models \varrho_j$, Item 1 of Definition 10 holds.

Let us now prove the two other items of Definition 10 related to injectivity. To do so, we prove a slightly stronger property than the second item: ($\star$) for all inj_k-event$(o, ev)^{\delta,\mu}$, $\mathrm{inj}_{k'}$-event$(o', ev')^{\delta',\mu'}$ occurring in Ψ, for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n), \tilde{\tau}' = (\tau'_1, \ldots, \tau'_n)$, for all substitutions σ, σ', if $k = k'$ and $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma')$ then for all $j_0 \in \{1, \ldots, n\}$, $F_{j_0} = \mathrm{inj}_{k_{j_0}}$-event$(o_{j_0}, ev_{j_0})$ implies $\tau_{j_0} = \tau'_{j_0}$.

We prove this property by contradiction. Hence let $\text{inj}_k\text{-event}(o, ev)^{\delta,\mu}$ and $\text{inj}_{k'}\text{-event}(o', ev')^{\delta',\mu'}$ occurring in Ψ such that $k = k'$ and $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma')$. Assume that there exists $j_0 \in \{1, \ldots, n\}$ such that $F_{j_0} = \text{inj}_{k_{j_0}}\text{-event}(o_{j_0}, ev_{j_0})$ and $\tau_{j_0} \neq \tau'_{j_0}$.

By Item 1 of Lemma 45, we know that for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash_{IO}^{n_{IO}} F_i\sigma$ and $T, \tau'_i \vdash_{IO}^{n_{IO}} F_i\sigma'$. By Lemma 2, we deduce from $\tau_{j_0} \neq \tau'_{j_0}$ that $o_{j_0}\sigma \neq o_{j_0}\sigma'$. Hence, $\mathsf{occ}_n(\bigwedge_{i=1}^n F_i\sigma) \neq \mathsf{occ}_n(\bigwedge_{i=1}^n F_i\sigma')$.

Moreover, by Lemma 45, we also know that there exist $(R = (H \wedge \phi \to C), \sigma_s, \varrho), (R' = (H' \wedge \phi' \to C'), \sigma'_s, \varrho') \in \mathcal{S}_{olj}$, two derivations $\mathcal{D}, \mathcal{D}'$ and four substitutions σ_ϱ, σ_d, σ'_ϱ, σ'_d such that:

- ϱ is of the form $\bigwedge_{i=1}^m G_i \Rightarrow \psi_\varrho$ with $m \geq n$ and $F_i\sigma = G_i\sigma_s\sigma_d$ for $i \leq n$;
- $\mathcal{D}$ is a derivation of $\curlywedge_{i=1}^m G_i\sigma_s\sigma_d$ at step $(\tau_{1,\varrho}, \ldots, \tau_{m,\varrho})$ from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}$ and $\tau_i = \tau_{i,\varrho}$ for $i \leq n$, $C = \curlywedge_{i=1}^m G_i\sigma_s$, $\sigma_d \vdash_i \phi$ and the outgoing edges of the root's child of $\mathcal{D}$ are labeled by $H\sigma_d$
- $T, \mu(\tilde{\tau}, \sigma) \vdash_i \text{inj}_k\text{-event}(o, ev)\sigma_\varrho$ and there exists $\text{inj}_k\text{-event}(o_1, ev_1) \in \psi_\varrho \cup \{G_i\}_{i=n+1}^m$ such that $\text{inj}_k\text{-event}(o, ev)\sigma_\varrho = \text{inj}_k\text{-event}(o_1, ev_1)\sigma_s\sigma_d$

and

- ϱ' is of the form $\bigwedge_{i=1}^{m'} G'_i \Rightarrow \psi'_\varrho$ with $m' \geq n$ and $F_i\sigma' = G'_i\sigma'_s\sigma'_d$ for $i \leq n$;
- $\mathcal{D}'$ is a derivation of $\curlywedge_{i=1}^{m'} G'_i\sigma'_s\sigma'_d$ at step $(\tau'_{1,\varrho}, \ldots, \tau'_{m',\varrho})$ from $\{R\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}'$ and $\tau'_i = \tau'_{i,\varrho}$ for $i \leq n$, $C' = \curlywedge_{i=1}^{m'} G'_i\sigma'_s$, $\sigma'_d \vdash_i \phi'$ and the outgoing edges of the root's child of $\mathcal{D}'$ are labeled by $H'\sigma'_d$
- $T, \mu'(\tilde{\tau}', \sigma') \vdash_i \text{inj}_{k'}\text{-event}(o', ev')\sigma'_\varrho$ and there exists $\text{inj}_{k'}\text{-event}(o'_1, ev'_1) \in \psi'_\varrho \cup \{G'_i\}_{i=n+1}^{m'}$ such that $\text{inj}_{k'}\text{-event}(o', ev')\sigma'_\varrho = \text{inj}_{k'}\text{-event}(o'_1, ev'_1)\sigma'_s\sigma'_d$.

We already proved that $\mathsf{occ}_n(\bigwedge_{i=1}^n F_i\sigma) \neq \mathsf{occ}_n(\bigwedge_{i=1}^n F_i\sigma')$. Hence since $F_i\sigma = G_i\sigma_s\sigma_d$ and $F_i\sigma' = G'_i\sigma'_s\sigma'_d$ for $i \leq n$, we deduce that $\mathsf{occ}_n(\bigwedge_{i=1}^m G_i\sigma_s\sigma_d) \neq \mathsf{occ}_n(\bigwedge_{i=1}^{m'} G'_i\sigma'_s\sigma'_d)$ and so if we denote $o_2 = \mathsf{occ}_n(\bigwedge_{i=1}^m G_i)$ and $o'_2 = \mathsf{occ}_n(\bigwedge_{i=1}^{m'} G'_i)$, we obtain that $o_2\sigma_s\sigma_d \neq o'_2\sigma'_s\sigma'_d$.

Since we proved that $T, \mu'(\tau'_1, \ldots, \tau'_n, \sigma') \vdash_i \text{inj}_{k'}\text{-event}(o', ev')\sigma'_\varrho$ and $T, \mu(\tau_1, \ldots, \tau_n, \sigma) \vdash_i \text{inj}_k\text{-event}(o, ev)\sigma_\varrho$ and we assumed that $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}', \sigma')$, we deduce from Lemma 2 that $o'\sigma'_\varrho = o\sigma_\varrho$. Furthermore, with $\text{inj}_{k'}\text{-event}(o', ev')\sigma'_\varrho = \text{inj}_{k'}\text{-event}(o'_1, ev'_1)\sigma'_s\sigma'_d$ and $\text{inj}_k\text{-event}(o, ev)\sigma_\varrho = \text{inj}_k\text{-event}(o_1, ev_1)\sigma_s\sigma_d$, we obtain that $o_1\sigma_s\sigma_d = o'_1\sigma'_s\sigma'_d$.

W.l.o.g., we can assume that the variables between (R, σ_s, ϱ) and $(R', \sigma'_s, \varrho')$ are disjoint. Hence, from $\sigma_d \vdash_i \phi$ and $\sigma'_d \vdash_i \phi'$, we deduce that $\sigma_d\sigma'_d \vdash_i \phi \wedge \phi'$. Moreover, since $o_2\sigma_s\sigma_d \neq o'_2\sigma'_s\sigma'_d$ and $o_1\sigma_s\sigma_d = o'_1\sigma'_s\sigma'_d$, we deduce that $\sigma_d\sigma'_d \vdash_i o'_1\sigma_s = o'_1\sigma'_s \wedge o_2\sigma_s \neq o'_2\sigma'_s$. Furthermore, $C\sigma_d\sigma'_d = C\sigma_d$ and $C'\sigma'_d = C'\sigma_d\sigma'_d$. Finally, the outgoing edges of the root's child of $\mathcal{D}$ (resp. $\mathcal{D}'$) are labeled by $H\sigma_d$ (resp. $H'\sigma'_d$) and so labeled by $H\sigma_d\sigma'_d$ (resp. $H'\sigma_d\sigma'_d$).

This allows us to build a derivation $\mathcal{D}_0$ of $(C \curlywedge C')\sigma_d\sigma'_d$ at step $(\tau_{1,\varrho}, \ldots, \tau_{m,\varrho}, \tau'_{1,\varrho}, \ldots, \tau'_{m',\varrho})$ from $\{R_0\}$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$ such that $T, \mathcal{S}_p, n_{IO} \vdash \mathcal{D}_0$ and $R_q = (H \wedge H' \wedge \phi \wedge \phi' \wedge o'_1\sigma_s = o'_1\sigma'_s \wedge o_2\sigma_s \neq o'_2\sigma'_s \to C \curlywedge C')$.

From Corollary 10, we know that for all $\varrho \in \mathcal{L}_i$, $(\vdash_{IO}^{n_{IO}}, \vdash_i, \mathsf{trace}_{IO}^{n_{IO}}(\mathcal{C}_I, \to_i)) \models \varrho$. Hence, we deduce that $\mathcal{H}yp_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(T, ())$ holds. Applying Theorem 4, we deduce that there exists an ordered derivation $\mathcal{D}'$ of $(C \curlywedge C')\sigma_d\sigma'_d$ at steps $(\tau_{1,\varrho}, \ldots, \tau_{m,\varrho}, \tau'_{1,\varrho}, \ldots, \tau'_{m',\varrho})$ from $\mathsf{saturateS}_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}^{\mathcal{S}_p}(\{R_q\}, \mathbb{C}_{sat})$ and $\mathbb{C}_{sat} \cup \mathbb{C}_e(T)$.

This is in contradiction with the fact that `verify_inj`$(\mathcal{S}_{olj}, n)$ terminates and is true which implies that $\mathsf{saturate}\mathsf{S}^{\mathcal{S}_p}_{\mathcal{L}\cup\mathcal{L}_i,\emptyset}(\{R_q\}, \mathbb{C}_{sat}) = \emptyset$. This allows us to deduce that for all $j_0 \in \{1, \ldots, n\}$, $F_{j_0} = \text{inj}_{k_{j_0}}\text{-event}(o_{j_0}, ev_{j_0})$ implies $\tau_{j_0} = \tau'_{j_0}$.

Notice that the second item of Definition 10 is directly implied by $(\star)$. Hence it remains to prove the last item of Definition 10. Consider $\text{inj}_k\text{-event}(o, ev)^{\delta,\mu}$ and $\text{inj}_{k'}\text{-event}(o', ev')^{\delta',\mu'}$ occurring in Ψ such that $k = k'$. We first show that for all tuples of steps $\tilde{\tau} = (\tau_1, \ldots, \tau_n)$, for all substitutions σ, if $\mu(\tilde{\tau}, \sigma)$ and $\mu'(\tilde{\tau}, \sigma)$ are defined then $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}, \sigma)$. Since $\mu(\tilde{\tau}, \sigma)$ and $\mu'(\tilde{\tau}, \sigma)$ are defined, we deduce from Item 1 of Lemma 45 that for all $i \in \{1, \ldots, n\}$, $T, \tau_i \vdash^{n_{IO}}_{IO} F_i\sigma$. Hence, by Lemma 45, there exist $(R = (H \wedge \phi \to C), \sigma_s, \varrho) \in \mathcal{S}_{olj}$ and two substitutions σ_ϱ, σ_d such that:

- ϱ is of the form $\bigwedge_{i=1}^{m} G_i \Rightarrow \psi_\varrho$ with $m \geq n$ and $F_i\sigma = G_i\sigma_s\sigma_d$ for $i \leq n$;
- there exists $\text{inj}_k\text{-event}(o_1, ev_1)^{\delta_1} \in \psi_\varrho \cup \{G_i\}_{i=n+1}^{m}$ such that $T, \mu(\tilde{\tau}, \sigma) \vdash_i \text{inj}_k\text{-event}(o, ev)\sigma_\varrho$ and $\text{inj}_k\text{-event}(o, ev)\sigma_\varrho = \text{inj}_k\text{-event}(o_1, ev_1)\sigma_s\sigma_d$.
- there exists $\text{inj}_{k'}\text{-event}(o'_1, ev'_1)^{\delta'_1} \in \psi_\varrho \cup \{G_i\}_{i=n+1}^{m}$ such that $T, \mu'(\tilde{\tau}, \sigma) \vdash_i \text{inj}_{k'}\text{-event}(o', ev')\sigma_\varrho$ and $\text{inj}_{k'}\text{-event}(o', ev')\sigma_\varrho = \text{inj}_{k'}\text{-event}(o'_1, ev'_1)\sigma_s\sigma_d$.

However, since `verify_inj`$(\mathcal{S}_{olj}, n)$ terminates and is true and since $k = k'$, we know that $(o_1\sigma_s, ev_1\sigma_s) = (o'_1\sigma_s, ev'_1\sigma_s)$. Hence, $\text{inj}_k\text{-event}(o_1, ev_1)\sigma_s\sigma_d = \text{inj}_{k'}\text{-event}(o'_1, ev'_1)\sigma_s\sigma_d$ which implies $\text{inj}_{k'}\text{-event}(o', ev')\sigma_\varrho = \text{inj}_k\text{-event}(o, ev)\sigma_\varrho$ and so $o\sigma_\varrho = o'\sigma_\varrho$. Considering that $T, \mu'(\tilde{\tau}, \sigma) \vdash_i \text{inj}_{k'}\text{-event}(o', ev')\sigma_\varrho$ and $T, \mu(\tilde{\tau}, \sigma) \vdash_i \text{inj}_{k'}\text{-event}(o, ev)\sigma_\varrho$, we can apply Lemma 2 to obtain that $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}, \sigma)$.

W.l.o.g., it remains to consider the cases where $\mu(\tilde{\tau}, \sigma)$ is defined and $\mu'(\tilde{\tau}, \sigma)$ is not defined. The idea is build another annotated query conclusion Ψ' from Ψ by replacing all instances of $F^{\delta,\mu}$ in Ψ with $F^{\delta,\mu''}$ such that:

- if F is not an injective event then $\mu'' = \mu$
- if $F = \text{inj}_k\text{-event}(o, ev)$ then for all steps $\tau_1, \ldots, \tau_n$, for all substitutions σ, $\mu''(\tilde{\tau}, \sigma) = \mu(\tilde{\tau}, \sigma)$ when there exists $\text{inj}_{k'}\text{-event}(o', ev')^{\delta'',\mu}$ occurring in Ψ where $k = k'$ and $\mu(\tilde{\tau}, \sigma)$ is defined; $\mu''(\tilde{\tau}, \sigma)$ is not defined otherwise.

Note that Ψ' is well defined since we proved that for $\text{inj}_k\text{-event}(o, ev)^{\delta,\mu}$ and $\text{inj}_{k'}\text{-event}(o', ev')^{\delta',\mu'}$ occurring in Ψ, $k = k'$ and $\mu(\tilde{\tau}, \sigma)$, $\mu'(\tilde{\tau}, \sigma)$ being defined implies $\mu(\tilde{\tau}, \sigma) = \mu'(\tilde{\tau}, \sigma)$.

The annotated conclusion query Ψ' satisfies Item 1 of Definition 10 since $\mu'_{|\text{dom}(\mu)} = \mu$ and Ψ satisfies Item 1 of Definition 10. Moreover, thanks to Ψ satisfying the property $(\star)$, it is easy to show that Ψ' also satisfies it which implies Item 2 of Definition 10. Finally, Ψ' satisfies Item 3 of Definition 10 by construction. □