Indistinguishability Beyond Diff-Equivalence in ProVerif

February 4, 2023

Abstract

When formalising cryptographic protocols, privacy-type properties such as strong flavours of secrecy, anonymity or unlinkability, are often modelled by indistinguishability statements. Proving them is notoriously more challenging than trace properties which benefit from a well-established tool support today. State-of-the-art techniques often exhibit significant limitations, e.g., consider only a bounded number of protocol sessions, or prove *diff-equivalence*—a fined-grained, structure-guided notion of indistinguishability that commonly yields unnecessarily pessimistic analyses.

In this paper, we design, implement and evaluate the first general framework for proving indistinguishability properties, for an unbounded number of protocol sessions, going beyond the scope of diff-equivalence. For that we relax the structural requirements of ProVerif, a state-of-the-art tool, through a notion of *session decomposition*, intuitively allowing a dynamic restructuration of the proofs. We can then verify in a modular way various, more realistic models of indistinguishability such as may-testing equivalence, by exhibiting for each relation a sufficient condition on ProVerif's output ensuring that it holds. We implement our approach into a prototype and showcase the gain in scope through several case studies.

1 Introduction

Cryptographic protocols are distributed programs enforcing the security of sensitive communications between several heterogeneous entities. Due to their distributed nature and their execution over untrusted networks, they exhibit complex behaviours making their analysis prohibitively tedious. Security properties of interest commonly include *trace properties* such as (weak) secrecy or authentication, benefiting today from a strong automation support. One may typically cite the ProVerif [BSCS20] and Tamarin [BCD+19] tools, that are able to handle full-fledged industrial protocols such as, TLS 1.3 [BBK17, CHH+17, BCW22], the 5G standard [BDH+18] or Signal [KBB17, CGCG+18].

Some classical security notions are however modelled by more complex (hyper)properties including *indistinguishability*. They take the form of equivalence relations in concurrent process algebra such as the applied π-calculus [ABF17], and capture strong privacy-type properties, like anonymity or unlinkability. Consider for example the prototypical scenario of a Basic Access Control protocol (BAC): reading devices at an airport interact with RFID chips of travelers' e-passports. We consider a process $S(k) = P(k) \mid R(k)$, where P and R (left abstract here for simplicity) are, respectively, processes modelling a passport and a reader, that have agreed on an identity-related value k. The unlinkability of identities across multiple sessions is formalised by a statement:

$$!\,\mathsf{new}\ k; !\,S(k)\ \approx\ !\,\mathsf{new}\ k; S(k) \tag{1}$$

The ! (*replication*) models an unbounded number of copies of the subsequent process, new k indicates a fresh value k, and $\approx$ is an equivalence relation modelling the indistinguishability of two hypothetical scenarios, from the adversary's view. Rephrasing, (1) models the impossibility to observe a difference between situations where all passports are different (right-hand side), and where some of them may interact multiple times with readers (left-hand side). In particular, a tool-assisted proof would have to accommodate for replicated processes of significantly different structures.

Yet, most analysis techniques for such properties are either limited to a bounded number of copies of S as in [CKR19], or to *diff-equivalence* as in ProVerif, Tamarin or Maude-NPA [SEMM14]. Diff-equivalence is a fined-grained relation relying on strong structural assumptions on the processes to be proved equivalent, significantly hindering its application scope. It is typically insufficient to prove unlinkability statements such as (1), except using specialised tools dedicated to specific types of protocols [HBD19, BDM20]. Diff-equivalence based proofs unsuccessfully attempt to *statically* match each (duplicated) instance of P and R from the right process, each with a (fresh) indistinguishable instance from the left process. Successful proofs require *dynamic* extensions of this argument, where this matching is different for each execution of each instance of P and R. Such data-dependent arguments are arguably beyond the capabilities of most protocol analysers for an unbounded number of sessions.

Contributions We extend ProVerif to support modular proofs of may-testing and observational equivalences, and their pre-orders, for an unbounded number of sessions.

1. We introduce a notion of *session decomposition*, inspired from symmetry reductions in the bounded case [CKR19]. Used as a substitute to ProVerif's internal representation of processes, it weakens the tool's requirements on their structure, to express coarser-grained security relations.

2. We then adapt ProVerif's approach—namely, converting processes into a set of Horn clauses—to account for session decompositions. The resulting clauses are saturated by ProVerif's usual solver, and we exhibit sufficient conditions on the saturated output; each ensures one relation (may-testing, observational equivalence, pre-orders...) on the initial processes. This shows in particular the modularity of our notion of session decomposition in that it is not tied to a specific equivalence relation.

3. We propose for each of the above-mentioned conditions a general, semantic version, as well as a refinement better suited for automated verification. We then implement the overall procedure and use it to successfully verify various examples escaping the current scope of ProVerif.

Related Work Proofs of equivalence properties in security protocols roughly follow two trends, commented below.

On the one hand, proofs for a bounded number of sessions put an emphasis on precision and termination of the analysis at the cost of not supporting replication [CCK12, CDD17, CKR18]. Among these, the SAT-Equiv tool may notably yield proofs for an unbounded number of sessions occasionally, but also imposes syntactic restrictions on processes that ProVerif is not subject to. We also mention [CKR19], which inspired our notion of session decomposition: it develops symmetry-based techniques for trace equivalence that allows them to prove efficiently unlinkability properties. Our approach is however more widely applicable, due to our support for unbounded processes, but also through the multiple security relations we handle. However, by being only based on sufficient conditions, our prototype is currently unable to exhibit attack traces, i.e., to disprove equivalence.

On the other hand, for an unbounded number of sessions, most tools only support indistinguishability notions in the spirit of diff-equivalence, typically ProVerif [BSCS20], Tamarin [BCD+19] or Maude-NPA [SEMM14]. Although an extension of ProVerif partially relaxes the underlying structural requirements [CB13], it is subsumed by our current approach in terms of scope. [CB13] also yields algorithmically less efficient procedures on examples where the two approaches overlap: its procedure can be seen as a *static* enumeration of all potential session decompositions, followed by a tentative of proof for each of them. On the contrary, our native integration of session decompositions makes our single-proof procedure lighter to process. Other extensions allow some form of dynamic reasoning for protocols with synchronisation [BS16]. Annotating processes with synchronisation points thus makes proofs beyond diff-equivalence possible; this however requires manual annotations, is limited in scope (synchronisations are prohibited under replications), and requires synchronisation assumptions not arising in our framework.

An important related work is Ukano [HBD19], a frontend to an altered version of ProVerif exhibiting sufficient conditions under which the decision of trace equivalence can be reduced to a combination of diff-equivalence and trace properties. These conditions are however only valid for specific properties of a syntactically restricted class of protocols. In particular, although Ukano is applicable to most examples we specifically consider in this paper, our more general and modular approach significantly broadens the technique's scope by not being subject to such restrictions (more details discussed in Section 7.2). The approach has also been carried in Tamarin [BDM20] to analyse stateful protocols.

Contents

2 Unlinkability As a Motivating Example

We give an overview of our results by outlining a proof of unlinkability of a simplified BAC e-passport protocol in our model. This will give an insight of the proof techniques we develop in the subsequent technical sections.

Modelling We consider an access control protocol where a passport P communicates with a reader R, after a preliminary chip scanning allowing the reader to retrieve a passport secret k. Subsequent cryptographic operations are described by *function symbols* senc and sdec, verifying the identity $\mathsf{sdec}(\mathsf{senc}(x,y,z),z) \to x$. An expression $\mathsf{senc}(m,r,k)$ models the symmetric encryption of a message m using a key k and randomness r, whereas $\mathsf{sdec}(u,k)$ models the decryption of u with k. A minimal version of the BAC protocol [For04] can then be described as follows, first in informal Alice-Bob notation:

$$\begin{array}{lll} P \to R: & n & \\ R \to P: & \mathsf{senc}(n,r,k) & \text{received as } x \\ P \to R: & \mathsf{ok} & \text{if } \mathsf{sdec}(x,k) = n \\ & \mathsf{error} & \text{otherwise} \end{array}$$

That is, P sends a freshly generated nonce n to R in clear as a challenge, and awaits as a response an encryption of n under their shared secret k. Then, the result of the protocol (ok or error) is output. This protocol is assumed to be carried out over an untrusted network, compromised by an adversary that may intercept messages, but also replay, forge or inject them. The desired unlinkability property is then:

> *After an arbitrary number of sessions of the protocol, the adversary cannot infer whether a same passport took part to several of these sessions.*

This security property should hold despite the adversary being able, e.g., to change the recipient of some messages and observe the resulting final $\mathsf{ok}/\mathsf{error}$ message. Formally, the roles of P and R are represented by *processes* of the applied π-calculus, written:

$$\begin{array}{rl} P(k) = & \mathsf{new}\ n; \mathsf{out}(c,n); \mathsf{in}(c,x); \\ & \mathsf{if}\ \mathsf{sdec}(x,k) = n\ \mathsf{then}\ \mathsf{out}(c,\mathsf{ok})\ \mathsf{else}\ \mathsf{out}(c,\mathsf{error}) \\ R(k) = & \mathsf{new}\ r; \mathsf{in}(d,y); \mathsf{out}(d,\mathsf{senc}(y,r,k)); 0 \end{array}$$

Here the 0 indicates a terminated process. The instructions $\mathsf{in}(u,x)$ and $\mathsf{out}(u,v)$ model inputs and outputs on a communication channel u. The $\mathsf{new}\ n$ formalises the fresh generation of a name n, unknown to the adversary until output on the channel c. The formulation of unlinkability we consider is the *equivalence* $S_{\mathit{left}} \approx S_{\mathit{right}}$, with:

$$\begin{aligned} S_{\mathit{left}} &= !\, \mathsf{new}\ k; !\, (P(k) \mid R(k)) \\ S_{\mathit{right}} &= !\, \mathsf{new}\ k; (P(k) \mid R(k)) \end{aligned}$$

where $A \mid B$ is the parallel composition of A and B, and $!\,A$ is the parallel composition of an unbounded number of copies of A. The right side of the equivalence is a process modelling a scenario where all sessions involve different passports, whereas the left side may involve several times the same passports in different sessions. The equivalence relation $\approx$ may then be chosen among several common choices such as *observational equivalence* $\approx_o$ or *may-testing equivalence* $\approx_m$ to model adversarial indistinguishability.

Instrumentation The first step of our approach to prove $S_{\mathit{left}} \approx S_{\mathit{right}}$ is to convert processes into *instrumented processes.* This internal representation of ProVerif intuitively interprets parallel processes as indexed sequences of processes. E.g., the process S_{right} is translated into the form:

$$\{\!\{ !_i^{o_1[i]} P_{\mathit{right}} \}\!\} \mid \{\!\{ !_i^{o_2[i]} R_{\mathit{right}} \}\!\}$$

with P_{right} and R_{right} respective translations of P and R. The multiset notation $\{\!\{\cdot\}\!\}$ (here, only singletons) indicates a set of parallel processes with a similar structure. Processes with a different structure are put in different multisets separated by a | operator. This is what we call a *session decomposition* in this paper. Inside these multisets, an annotated replication $!_i^o P$ can be seen as a collection of copies of P, indexed by the variable i, and uniquely identified during the analysis by the *occurrence o*. In particular, P_{right} and R_{right} are mostly identical to P and R, except that they have their private names freshly renamed and indexed by i, e.g. n into $n_r[i]$. The argument i indicates a dependency, expressing that $n_r[i]$ is a different fresh name for each different copy of the process (i.e., for each instance of i). This thus makes the new instruction superfluous. For example:

$$R_{\mathit{right}} = \mathsf{in}^{o_r[i]}(d, y); \mathsf{out}(d, \mathsf{senc}(y, r_r[i], k_r[i])); 0$$

Again, the occurrence $o_r[i]$ is used for making reference to the tagged instruction (here, the input) during the analysis later. So far, most of this material exists in ProVerif; the novel ingredient is that, when translating S_{left}, we obtain a session decomposition similar to the previous one, despite the different initial number of replications:

$$\{\!\{ !_{i,j}^{o_3[i,j]} P_{\mathit{left}} \}\!\} \mid \{\!\{ !_{i,j}^{o_4[i,j]} R_{\mathit{left}} \}\!\}$$

The process S_{left} can indeed also be seen as a collection of copies of P and R, albeit for different data dependencies. This is reflected in this session decomposition by the double index $[i, j]$, where i indexes to the first replication and j the second. The indexation of R is therefore this time:

$$R_{\mathit{left}} = \mathsf{in}^{o_\ell[i,j]}(d, y); \mathsf{out}(d, \mathsf{senc}(y, r_\ell[i, j], k_\ell[i])); 0$$

In short, instead of requiring that two equivalent processes have (syntactically) the same structure, our analysis criterion is based on the components (multisets) of the session decomposition, regardless of their sizes. Theorem 1 shows that instrumentation preserves process equivalences.

Translation Into Clauses Once the session decomposition is over, the algorithm translates the instrumented processes into Horn clauses (Section 5):

$$F_1 \wedge \cdots \wedge F_n \rightarrow F$$

These clauses intuitively model ProVerif's internal reasoning and include atomic formulae of various forms. Some clauses, already present in the baseline version of ProVerif, describe for example the adversary's capabilities, such as:

$$\mathrm{input}(x, y) \wedge \mathrm{msg}(x, z, y', z') \wedge y \neq y' \rightarrow \mathrm{bad}$$

The fact $\mathrm{input}(x, y)$ models that the attacker can listen on the channel x in an execution of S_{left}, and y in an equivalent execution of S_{right}. The clause represents the fact that the attacker can distinguish the two executions (fact bad) when a message was sent on x in S_{left} but on a different channel y' in S_{right} (fact $\mathrm{msg}(x, z, y', z')$). Other facts $\mathrm{ev}(e, e')$ indicates that an *event* e occurs in S_{left}, and simultaneously e' occurs in S_{right}. Also, importantly, $\mathrm{att}(u, v)$ indicates that the adversary is able to compute u (resp. v) using the knowledge accumulated in S_{left} (resp. S_{right}) by intercepting outputs. Finally, we introduce in this work novel events $\mathit{repl}(o)$ (*replication events*), indicating that a replication labelled with the occurrence o is unfolded. We typically use them to express correspondences between occurrences (e.g., o_3 and o_1 in the example), which in turn helps capturing our notion of session decomposition within clauses. For example, the following clause describes the execution of the first output of P in S_{left} and S_{right}

$$\mathrm{ev}(\mathit{repl}(o_3[i, j]), \mathit{repl}(o_1[i'])) \to \mathrm{att}(n_l[i, j], n_r[i'])$$

Rephrasing, when an occurrence of the replication $o_3[i, j]$ is executed in S_{left}, while $o_1[i']$ is executed in S_{right}, the adversary learns the corresponding values of n (as they are publicly output). Note in particular that the premise of the clause relates events occurring at different structural levels of processes (i.e., o_3 is under two nested replications, unlike o_1), which is not a natural feature of diff-equivalence.

Verification After generating all clauses, they are saturated using ProVerif's internal solver. When none of the saturated clauses concludes bad, we can directly conclude that diff-equivalence holds, as in ProVerif. The key novelty of our approach is that even when some saturated clauses conclude bad, we may still be able to prove equivalences as follows.

Intuitively, a clause $H \to \mathrm{bad}$ indicates that there may be distinguishable executions of S_{left} and S_{right} that satisfy H. Conversely, the *saturation* procedure of ProVerif ensures that if some executions of S_{left} and S_{right} are distinguishable then there must exist a saturated clause $H \to \mathrm{bad}$ where H is satisfied by these executions (Theorem 2). Thus, to prove may-testing inclusion $\sqsubseteq_m$ for example, it suffices to build a *session matching* (Section 6.2) mapping the replication events of any execution of S_{left} to the replication occurrences of S_{right}, that can *falsify* (Section 6.3) the hypotheses H of all saturated clauses $H \to \mathrm{bad}$. In our example, it means for instance mapping instantiations of $\mathit{repl}(o_3[i, j])$ to adequate instantiations of $o_1[i']$.

In other words, our conditions implies that all executions of S_{left} can be matched by an execution of S_{right} that satisfy none of the hypotheses of the clauses concluding bad, which thus entails that these two executions are not distinguishable (Theorem 3). We exhibit similar conditions implying observational equivalence $\approx_o$ and its pre-order $\sqsubseteq_o$ (simulation). In our implementation, we can let the prototype find the "best" (i.e., finest) proofs it can. In the case of this example, the tool automatically shows that $S_{\mathit{left}} \sqsubseteq_m S_{\mathit{right}}$ and $S_{\mathit{right}} \sqsubseteq_o S_{\mathit{left}}$. Although not formalised by the tool as it does not compute attacks, we can show by hand that the tool's result is optimal, i.e., $S_{\mathit{right}} \not\sqsupseteq_o S_{\mathit{left}}$.

3 Model

In this section, we present the process calculus and the notions of equivalence studied in this paper. Most of it is standard material from the theory of ProVerif.

3.1 Syntax

We assume a classical term algebra, that is, an infinite set of *variables* $\mathcal{V}$, an infinite set of *names* $\mathcal{N} = \mathcal{N}_{\text{pub}} \uplus \mathcal{N}_{\text{prv}}$ representing atomic values (public and private, respectively), and a finite set of *function symbols* (with their arity) called a *signature*. Specifically, we consider two distinct sets $\mathcal{F}_d$ and $\mathcal{F}_c$ representing, respectively, *constructor* symbols (used to build messages, e.g., encryption or concatenation) and *destructor* symbols (representing operations on messages that may fail, e.g., decryption). Functions, names and variables can be combined to form *expressions*:

$$\begin{array}{rll} D ::= & a & \textit{atomic value } a \in \mathcal{N} \cup \mathcal{V} \\ & f(D_1, \ldots, D_k) & \textit{application } f \in \mathcal{F}_d \cup \mathcal{F}_c \\ & \mathsf{fail} & \textit{failure} \end{array}$$

An expression is called a *term* when it does not contain destructors or the fail symbol, and is then referred by the grammar tokens M, N. Terms are therefore expressions that correspond, intuitively, to successful computations. We then consider *processes* modelling concurrent programs:

$$\begin{array}{rll} P, Q ::= & 0 & \textit{nil} \\ & \mathsf{out}(N, M); P & \textit{output} \\ & \mathsf{in}(N, x); P & \textit{input } (x \in \mathcal{V}) \\ & \mathsf{event}(M); P & \textit{event} \\ & P \mid Q & \textit{parallel composition} \\ & !\, P & \textit{replication} \\ & \mathsf{new}\ n; P & \textit{restriction } (n \in \mathcal{N}) \\ & \mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q & \textit{assignment} \end{array}$$

We already discussed most of these constructions during the motivation example, at the exception of events. Events are ignored during equivalence proofs, although they can be used in ProVerif to specify *axioms*, introduced in [BCC22], that are trace properties guiding the internal decision procedure. They are admitted during verification, but those mentioned and introduced in this paper are (protocol-independent) properties that have been priorly proved manually.

The assignment instruction $\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q$ attempts to evaluate D and executes $P(x)$ with the resulting value x in case of success, or executes a default process Q in case of failure. Assuming a symbol $\mathsf{Equals} \in \mathcal{F}_d$ with the rewrite rule $\mathsf{Equals}(x, x) \to \mathsf{ok}$, assignments can therefore be used to encode conditionals as in the motivation example.

The notion of evaluation underlying assignments is formalised through a set of *rewrite rules* $\ell \to r$, such as the rule $\mathsf{sdec}(\mathsf{senc}(x, y, z), z) \to x$ used in the motivation example. Formally, we refer to the usual notion of substitutions $\sigma = \{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\}$, i.e., we write $M\sigma$ the term where all syntactic occurrences of x_i in M are replaced by M_i. We naturally extend the application of a substitution to expressions, processes, etc. ProVerif then operates using the *evaluation* of expressions $D \Downarrow U$ where U is either a term M or the constant fail. This evaluation is based on a standard notion of rewriting system normalising expressions with a call-by-value strategy (full details in Appendix **??**).

3.2 Operational Semantics

The semantics of processes characterises their behaviour when executed in a hostile environment modelling an adversary controlling the communication network. It operates on *config-*

urations $\mathcal{P}, \Phi$, which are tuples where $\mathcal{P}$ is a multiset of processes modelling all processes currently executed in parallel, and Φ is a called a *frame*, i.e., a substitution:

$$\Phi = \{{}^{M_1}/_{\omega_1}, \ldots, {}^{M_n}/_{\omega_n}\}$$

Intuitively, a frame records the knowledge obtained by the adversary by spying on outputs sent on the communication network. An entry ${}^{M_i}/_{\omega_i}$ indicates in particular that the adversary can access the value of M_i through the *handle* ω_i, which is a dedicated type of variable. In particular, the following notion formalises attacker's computations:

Definition 1 (Recipe)**.** *A* recipe ξ *is an expression without private names, and with no variables except handles.*

For example, a frame $\Phi = \{{}^{\mathsf{senc}(m,k)}/_{\omega_1}, {}^{k}/_{\omega_2}\}$ with $m, k \in \mathcal{N}_{\mathrm{prv}}$ indicates that the attacker observed, successively, $\mathsf{senc}(m, k)$ and k. The term m can be thus computed by the adversary using the recipe $\xi = \mathsf{sdec}(\omega_1, \omega_2)$; that is, $\xi\Phi \Downarrow m$. Note in particular that the decryption key k is not used directly (as $\mathsf{sdec}(\omega_1, k)$ is not a valid recipe because it contains the private name k), but by reference through the handle ω_2. The actual semantics is then defined in Figure 1, as a labelled transition relation $\xrightarrow{\alpha}$ over configurations, where the label α is called an *action*:

- $\mathsf{in}(\xi, \zeta)$ materialises an input fetched from the adversary, where the input term is computed by the recipe ζ, and the underlying channel N is public in that it can be computed by the adversary through recipe ξ;

- $\mathsf{out}(\xi, \omega)$ materialises an output sent on a channel publicly computable through ξ, and added to the adversary's knowledge as a frame entry at handle ω;

- and finally, events make the corresponding term appear as the transition label, and empty labels are used for miscellaneous rules without visible behaviours.

$$\{\!\{0\}\!\}, \Phi \to \emptyset, \Phi \qquad (\textsc{Nil})$$

$$\{\!\{P \mid Q\}\!\}, \Phi \to \{\!\{P, Q\}\!\}, \Phi \qquad (\textsc{Par})$$

$$\{\!\{!P\}\!\}, \Phi \to \{\!\{P, !P\}\!\}, \Phi \qquad (\textsc{Repl})$$

$$\{\!\{\mathsf{new}\ a; P\}\!\}, \Phi \to \{\!\{P\{{}^{a'}/_{a}\}\}\!\}, \Phi \quad \text{if } a' \in \mathcal{N}_{\mathrm{prv}} \smallsetminus \mathit{names}(\mathcal{P}, P, \Phi) \qquad (\textsc{Restr})$$

$$\{\!\{\mathsf{out}(N, M); P, \mathsf{in}(N, x); Q\}\!\}, \Phi \to \{\!\{P, Q\{{}^{M}/_{x}\}\}\!\}, \Phi \qquad (\textsc{Comm})$$

$$\{\!\{\mathsf{out}(N, M); P\}\!\}, \Phi \xrightarrow{\mathsf{out}(\xi,\omega)} \{\!\{P\}\!\}, \Phi \cup \{{}^{M}/_{\omega}\} \quad \text{if } \omega \notin \mathit{dom}(\Phi),\ \xi\Phi \Downarrow N \qquad (\textsc{Out})$$

$$\{\!\{\mathsf{in}(N, x); Q\}\!\}, \Phi \xrightarrow{\mathsf{in}(\xi,\zeta)} \{\!\{Q\{{}^{M}/_{x}\}\}\!\}, \Phi \quad \text{if } \xi, \zeta \text{ recipes such that } \xi\Phi \Downarrow N \text{ and } \zeta\Phi \Downarrow M \qquad (\textsc{In})$$

$$\{\!\{\mathsf{event}(M); P\}\!\}, \Phi \xrightarrow{M} \{\!\{P\}\!\}, \Phi \qquad (\textsc{Event})$$

$$\{\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \Phi \to \{\!\{R\}\!\}, \Phi \quad \text{if } R = P\{{}^{M}/_{x}\} \text{ if } D \Downarrow M;\ R = Q \text{ if } D \Downarrow \mathsf{fail} \qquad (\textsc{Let})$$

$$\mathcal{P} \cup \mathcal{Q}, \Phi \xrightarrow{\alpha} \mathcal{P}' \cup \mathcal{Q}, \Phi' \qquad \text{if } \mathcal{P}, \Phi \xrightarrow{\alpha} \mathcal{P}', \Phi' \qquad (\textsc{Cont})$$

Figure 1: Operational Semantics Between Configurations

Definition 2 (Trace)**.** *A* trace T *of a configuration* $\mathcal{C}_0$ *is a sequence of transitions of Figure 1 starting from* $\mathcal{C}_0$*, written* $T : \mathcal{C}_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} \mathcal{C}_n$*. We may write* $\mathcal{C}_0 \xRightarrow{\alpha_1 \cdots \alpha_n} \mathcal{C}_n$ *when intermediary processes are not relevant. Note that in the work* $\alpha_1 \cdots \alpha_n$*, empty* α_i*, i.e., transitions without labels, are implicitly omitted. By extension, we may consider traces of a process* P *by interpreting* P *as the configuration* $\{\!\{P\}\!\}, \emptyset$*.*

3.3 Security Properties

We now define the notions of process equivalence considered in this paper. They are typically used to model strong flavours of privacy-type properties of cryptographic protocols expressed in our process algebra. They are intuitively built over a notion of *static indistinguishability* that, intuitively, formalises that the adversary cannot compute a test that tells two given history of observables apart. In the following, we implicitly assume that the signature includes a symbol binary symbol Equals defined by $\mathsf{Equals}(x, x) \to \mathsf{ok}$ that allows the adversary to run equality tests to violate indistinguishability. This static notion is then lifted to dynamic behaviours through the operational semantics. For that we consider a relation over words actions: we write

$$\alpha_1 \cdots \alpha_n \equiv \beta_1 \cdots \beta_p$$

if the two words $\alpha_1 \cdots \alpha_n$ and $\beta_1 \cdots \beta_p$ become identical after removing all event actions from them. Not considering event actions in the definition of equivalences models that they are rather annotations from the modeller than observables of the attacker. They are, instead, convenient to express axioms, that take in this paper the form of trace properties.

Definition 3 (Observational equivalence)**.** Observational pre-order *(or* simulation*)* $\sqsubseteq_o$ *is defined as the largest relation* $\mathcal{R}$ *over configurations such that* $\mathcal{C}\ \mathcal{R}\ \mathcal{C}'$ *implies, if we write* $\mathcal{C} = \mathcal{P}, \Phi$ *and* $\mathcal{C}' = \mathcal{P}', \Phi'$*:*

1. *for all recipes* ξ*,* $\xi\Phi \Downarrow \mathsf{fail}$ *iff* $\xi\Phi' \Downarrow \mathsf{fail}$*;*
2. *if* $\mathcal{C} \xrightarrow{\alpha} \mathcal{C}_1$ *then there exists* $\mathcal{C}'_1$ *such that* $\mathcal{C}' \overset{w}{\Rightarrow} \mathcal{C}'_1$*,* $\alpha \equiv w$*, and* $\mathcal{C}_1\ \mathcal{R}\ \mathcal{C}'_1$*.*

Observational equivalence $\approx_o$ *is defined as the largest* symmetric *relation* $\mathcal{R}$ *that satisfies properties 1, 2.*

The usual definition of observational equivalence is context-based but often harder to handle in proofs. The above definition is a standard, more operational but equivalent characterisation called bisimilarity [ABF17]. The following, coarser notion of equivalence, *may testing*, and corresponds to the indistinguishability of sets of traces. It intuitively states for any trace of either process, and for any computation the adversary may additionally do, an indistinguishable sequence of actions can be taken in the other process.

Definition 4 (May testing)**.** May testing inclusion *between configurations, denoted* $\mathcal{C} \sqsubseteq_m \mathcal{C}'$*, holds when for all traces* $\mathcal{C} \overset{w}{\Rightarrow} (\mathcal{P}, \Phi)$ *and for all sets of recipes* S*, there exists a trace* $\mathcal{C}' \overset{w'}{\Longrightarrow} (\mathcal{P}', \Phi')$ *such that* $w \equiv w'$ *and for all* $\xi \in S$*,* $\xi\Phi \Downarrow \mathsf{fail}$ *iff* $\xi\Phi' \Downarrow \mathsf{fail}$*.* May testing equivalence*, denoted* $\approx_m$*, is defined as* $\sqsubseteq_m \cap \sqsupseteq_m$*.*

4 Instrumentation

Recalling the motivation example, the first step of our procedure is to convert processes into *instrumented* ones that record some data dependencies, and more importantly, compute *session decompositions* to materialise the internal symmetries of processes.

4.1 Instrumented Processes

We introduce the set $\mathcal{X}_\lambda$ of *session variables*, used to index replicated data, and are instantiated into *session identifiers*. Names are thus now represented by *name patterns*

$$\overline{n} = n[\mathsf{a}_1, \dots, \mathsf{a}_k]$$

where each a_i is called an *argument pattern*, that is either a term (representing a prior input), or a session variable or identifier (representing a replication in scope). Public names a are implicitly interpreted as name patterns $a[]$. We define below the grammar of such instrumented processes.

$$\begin{array}{lll} P, Q ::= & 0 & \mathsf{in}^{\overline{n}}(N, x); P \\ & P \mid Q & \{\!\!\{ !^{\overline{n}_1}_{\tilde{\mathsf{a}}_1} P_1; \dots ; !^{\overline{n}_k}_{\tilde{\mathsf{a}}_k} P_k \}\!\!\} \\ & \mathsf{event}(M); P & \mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q \\ & \mathsf{out}(N, M); P & \end{array}$$

The major addition compared to regular processes is the *session decomposition* $\{\!\!\{ !^{\overline{n}_1}_{\tilde{\mathsf{a}}_1} P_1; \dots ; !^{\overline{n}_k}_{\tilde{\mathsf{a}}_k} P_k \}\!\!\}$. Intuitively, it gathers processes $P_1, \dots, P_k$ of same structure, whose replicated copies will be indexed by the session variables in $\tilde{\mathsf{a}}_1, \dots, \tilde{\mathsf{a}}_k$, recalling the motivating example. This notation extends to non-replicated processes P_i with $\tilde{\mathsf{a}}_i = \emptyset$.

The task of proving the equivalence of two processes P and Q rephrases, intuitively, to matching the observables (inputs and outputs) of P with those of Q. This is done mostly syntactically in ProVerif while, for session decompositions:

$$P = \{\!\!\{ !^{\overline{n}_1}_{\tilde{\mathsf{a}}_1} P_1; \dots ; !^{\overline{n}_k}_{\tilde{\mathsf{a}}_k} P_k \}\!\!\} \qquad Q = \{\!\!\{ !^{\overline{m}_1}_{\tilde{\mathsf{b}}_1} Q_1; \dots ; !^{\overline{m}_\ell}_{\tilde{\mathsf{b}}_\ell} Q_\ell \}\!\!\}$$

our approach may match any observable from a replicated copy of P_i with an equivalent one from a copy of Q_j. Roughly, the current procedure of ProVerif can only handle, in comparison, the case $k = \ell$ and may only match P_i with Q_i. ProVerif also has a requirement that P and Q have the same structure, that we also extend to our context. In the above case, this (recursively) means that all P_i, Q_j have the same structure, and that there are the same ordinal number of sessions on each side (non replicated processes, i.e., $\tilde{\mathsf{a}}_i = \emptyset$, counting for 1, and replicated processes, i.e., $\tilde{\mathsf{a}}_i \neq \emptyset$, for $+\infty$). In the following definition, we refer in particular as $\#\tilde{\mathsf{a}}_i \in \{1, +\infty\}$ to the corresponding number of sessions, also using the natural extension of addition to $\mathbb{N} \cup \{+\infty\}$.

Definition 5 (Control-flow equivalence)**.** Control flow equivalence $\approx_{cf}$ *is the smallest relation on instrumented processes such that:*

- $\{\!\!\{ !^{\bar{n}_1}_{\tilde{\mathsf{a}}_1} P_1, \dots, !^{\bar{n}_k}_{\tilde{\mathsf{a}}_k} P_k \}\!\!\} \approx_{cf} \{\!\!\{ !^{\bar{m}_1}_{\tilde{\mathsf{b}}_1} Q_1, \dots, !^{\bar{m}_l}_{\tilde{\mathsf{b}}_l} Q_l \}\!\!\}$ *if for all i, j, $P_i \approx_{cf} Q_j$ and $\sum_{i=1}^{k} \#\tilde{\mathsf{a}}_i = \sum_{j=1}^{l} \#\tilde{\mathsf{b}}_j$;*
- *let any instructions $\alpha[P_1, \dots, P_n]$ and $\beta[Q_1, \dots, Q_n]$ of the same type (nil, input, output, event, parallel, let; thus $n \in [\![0, 2]\!]$). If for all $i \in [\![1, n]\!]$, $P_i \approx_{cf} Q_i$, then $\alpha \approx_{cf} \beta$.*

4.2 From Processes to Instrumented Processes

We now detail how to transform a regular process into an instrumented one. Up to alpha renaming, we assume without loss of generality that processes bind names and variables at most once (by in or new instructions). Most of the instrumentation process intuitively consists of tagging inputs and replications by fresh occurrences to identify them, and to record the tags in scope in the replication's arguments. Such a procedure already exists in ProVerif as detailed in [BCC22], and is given in Figure 2. We take below a particular focus on the new material (session matchings). The transformation takes the form of a ternary relation between processes $P \Downarrow_{\tilde{\mathsf{a}}} Q$, where P is a process, Q is an instrumented process and $\tilde{\mathsf{a}}$ is a sequence of argument patterns intuitively recording the data dependencies in scope. Typically, replications are instrumented as follows:

$$!\,P \ \Downarrow_{\tilde{\mathsf{a}}} \ \{\!\!\{ !^{o[\tilde{\mathsf{a}},i]}_{\{i\}} P' \}\!\!\}$$

where, for some fresh occurrence o and session variable $i \in \mathcal{X}_\lambda$, $P \Downarrow_{\tilde{\mathsf{a}}\cdot i} P'$. This means that i serves as a fresh placeholder for indexing the various copies of P. It is added to the record $\tilde{\mathsf{a}}$ for the next steps of the instrumentation $\Downarrow_{\tilde{\mathsf{a}}\cdot i}$, and to the dependencies of the fresh occurrence label o. The tagging is similar for inputs:

$$\mathsf{in}(N,x);P \ \Downarrow_{\tilde{\mathsf{a}}} \ \mathsf{in}^{o[\tilde{\mathsf{a}}_{|\lambda}]}(N,x);P'$$

with some fresh o and if we have $P \Downarrow_{\tilde{\mathsf{a}}\cdot x} P'$, and $\tilde{\mathsf{a}}_{|\lambda}$ refers to the sequence $\tilde{\mathsf{a}}$ with input terms removed, thus keeping only session variables/identifier. Only inputs and replications (i.e., the sources of unboundedness) need to be tagged with occurrences, meaning that $\Downarrow_{\tilde{\mathsf{a}}}$ is extended to most other process constructions (inputs, outputs, events, let) in the natural way, without modification. The only exception is the parallel operator which, in the same way as replication, has to produce a session matching:

$$P \mid Q \ \Downarrow_{\tilde{\mathsf{a}}} \ \{\!\!\{ !^{o[\tilde{\mathsf{a}}]}_{\emptyset} P' \}\!\!\} \mid \{\!\!\{ !^{o'[\tilde{\mathsf{a}}]}_{\emptyset} Q' \}\!\!\}$$

with some fresh o, o' and if we have $P \Downarrow_{\tilde{\mathsf{a}}} P'$, $Q \Downarrow_{\tilde{\mathsf{a}}} Q'$. This session matching is "blank" at the moment, that is, it does not record any potential structural symmetries between P and Q.

$$0 \ \Downarrow_{\tilde{\mathsf{a}}} \ 0$$

$$\mathsf{out}(N,M);P \ \Downarrow_{\tilde{\mathsf{a}}} \ \mathsf{out}(N,M);P' \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}} P'$$

$$\mathsf{in}(N,x);P \ \Downarrow_{\tilde{\mathsf{a}}} \ \mathsf{in}^{o[\tilde{\mathsf{a}}_{|\lambda}]}(N,x);P' \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}\cdot x} P' \text{ and } o \in \mathcal{N}_{\mathsf{in}} \text{ fresh}$$

$$\mathsf{event}(M);P \ \Downarrow_{\tilde{\mathsf{a}}} \ \mathsf{event}(M);P' \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}} P'$$

$$\mathsf{new}\ n;P \ \Downarrow_{\tilde{\mathsf{a}}} \ P' \qquad \text{if } P\{^{n'[\tilde{\mathsf{a}}]}/_{n}\} \Downarrow_{\tilde{\mathsf{a}}} P' \qquad \text{for some fresh name pattern } n'$$

$$\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q \ \Downarrow_{\tilde{\mathsf{a}}} \ \mathsf{let}\ x = D\ \mathsf{in}\ P'\ \mathsf{else}\ Q' \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}} P' \text{ and } Q \Downarrow_{\tilde{\mathsf{a}}} Q'$$

$$P \mid Q \ \Downarrow_{\tilde{\mathsf{a}}} \ \{\!\!\{ !^{o[\tilde{\mathsf{a}}]}_{\emptyset} P' \}\!\!\} \mid \{\!\!\{ !^{o'[\tilde{\mathsf{a}}]}_{\emptyset} Q' \}\!\!\} \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}} P',\ Q \Downarrow_{\tilde{\mathsf{a}}} Q', \text{ and } o, o' \in \mathcal{N} \text{ are fresh}$$

$$!P \ \Downarrow_{\tilde{\mathsf{a}}} \ \{\!\!\{ !^{o[\tilde{\mathsf{a}},i]}_{\{i\}} P' \}\!\!\} \qquad \text{if } P \Downarrow_{\tilde{\mathsf{a}}\cdot i} P', \text{ and } o \in \mathcal{N}, i \in \mathcal{X}_\lambda \text{ fresh}$$

Figure 2: Transformation into Instrumented Processes

Computing symmetries is managed by the second part of the transformation, taking the form of three *factorisation rules* $\rightsquigarrow$ that normalise the structure of session decompositions.

We recall the intuition that $P \mid Q$ models two arbitrary parallel processes P and Q, while $\{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} P, !^{\bar{m}}_{\tilde{\mathsf{b}}} Q \}\!\!\}$ carries additional information about structural symmetries (i.e., $P \approx_{cf} Q$). These factorisation rules intuitively merge parallel processes into multisets when structural symmetries are identified. All rules are to be understood up to associativity and commutativity of the parallel operator, and are applied to any subprocesses of the instrumented process:

$$\begin{aligned}
\{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} (P \mid Q) \}\!\!\} \cup S &\rightsquigarrow \{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} P \}\!\!\} \mid \{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} Q \}\!\!\} \mid S \\
\{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} \{\!\!\{ !^{\bar{n}_i}_{\tilde{\mathsf{a}}_i} P_i \}\!\!\}_{i=1}^{n} \}\!\!\} \cup S &\rightsquigarrow \{\!\!\{ !^{\bar{n}_i}_{\tilde{\mathsf{a}} \cup \tilde{\mathsf{a}}_i} P_i \}\!\!\}_{i=1}^{n} \mid S \\
\{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} P \}\!\!\} \cup S \mid \{\!\!\{ !^{\bar{m}}_{\tilde{\mathsf{b}}} Q \}\!\!\} \cup S' &\rightsquigarrow \{\!\!\{ !^{\bar{n}}_{\tilde{\mathsf{a}}} P, !^{\bar{m}}_{\tilde{\mathsf{b}}} Q \}\!\!\} \cup S \cup S' \\
&\qquad\qquad\qquad \text{if } P \approx_{cf} Q
\end{aligned}$$

The first two rules make replication and parallel operators collapse, to present them under a compact normalised form that facilitates the search for structural symmetries. Note however that these two operations technically break priorly established symmetries, which is why applying them replaces the multiset union ($\cup$) by a regular parallel composition ($\mid$). On the contrary, the last rule exhibits symmetries, and records them by merging session decompositions whose base processes P, Q are control-flow equivalent.

Definition 6 (Instrumentation)**.** *Given a process P, we write $[\![P]\!]_i$ to refer to an* instrumentation *of P, i.e., an instrumented process such that $P \Downarrow_{\emptyset} \rightsquigarrow^{*} [\![P]\!]_i \not\rightsquigarrow$.*

Example 1. We already illustrated in the motivation example how our notion of session decomposition helped treating more processes as control-flow equivalent in comparison with the usual approach of ProVerif. One step further, two processes whose syntax are significantly different, e.g., $(!\,P) \mid Q \mid (!\,\mathsf{new}\ n; !\,R)$ and $!\,S$, have instrumentations that may effectively be control-flow equivalent, when those of P, Q, R and S are.

4.3 Instrumented semantics

On single processes We now adapt the operational semantics to instrumented processes. Intuitively, it is a decision-oriented version of the operational semantics: in addition to the explicit replication labels, the attacker's operations are also represented explicitly, facilitating their encoding into Horn clauses for decisional purposes. As such, the semantics operates on *instrumented configurations* $\mathcal{P}, \mathcal{A}, \Lambda$, where $\mathcal{P}$ is a multiset of instrumented processes, $\mathcal{A}$ is a set of terms representing the attacker's knowledge and Λ is a set of name patterns tracking unfolded replications. The semantics is formalised in Figure 3.

For example Rule I-App evidences that M is computable by the adversary, and adds it to the knowledge base $\mathcal{A}$. The adversary may also introduces its own (fresh) constants in computations, which take the form of name patterns $b_0[\lambda]$, with b_0 a fixed (but fresh) name and λ a session identifier, through Rule I-Gen. Some terms $M, N \in \mathcal{A}$ may then later be used by, e.g., Rules I-In and I-Out. One other important rule is I-Repl handling both replication and parallel composition, spawning a copy of a process with a fresh occurrence. In case of a non-replicated process, i.e., when $\tilde{\mathsf{a}} = \emptyset$, the last condition notably prevents it from being replicated more than once. Finally, we also mention that the rule for inputs notably triggers a *precise event* $pre(\bar{o}, M)$; some internal *axioms* of ProVerif make references to such precise events to guide, and thus improve the precision of, the decision procedure. They are, in this paper, manually-proved trace properties that are protocol-independent. We use in particular a set of so-called *precise axioms*, described and implemented in [CCT18, BCC22].

$$\{\!\{0\}\!\}, \mathcal{A}, \Lambda \to_i \emptyset, \mathcal{A}, \Lambda \qquad \text{(I-N}\textsc{il})$$

$$\{\!\{P \mid Q\}\!\}, \mathcal{A}, \Lambda \to_i \{\!\{P, Q\}\!\}, \mathcal{A}, \Lambda \qquad \text{(I-P}\textsc{ar})$$

$$\{\!\{\{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}}P\}\!\} \cup \mathcal{M}\}\!\}, \mathcal{A}, \Lambda \xrightarrow{repl(\overline{o}\sigma)}_i \{\!\{P\sigma, \{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}}P\}\!\} \cup \mathcal{M}\}\!\}, \mathcal{A}, \Lambda \cup \{\overline{o}\sigma\}$$
$$\text{if } dom(\sigma) = \tilde{\mathsf{a}},\ img(\sigma) \subseteq \mathbb{N},\ \overline{o}\sigma \notin \Lambda \qquad \text{(I-R}\textsc{epl})$$

$$\{\!\{\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\}\!\}, \mathcal{A}, \Lambda \to_i \{\!\{R\}\!\}, \mathcal{A}, \Lambda$$
$$\text{with } R = \{\!\{P\{^M/_x\}\}\!\} \text{ if } D \Downarrow M \text{ and } R = Q \text{ if } D \Downarrow \mathsf{fail} \qquad \text{(I-L}\textsc{et})$$

$$\{\!\{\mathsf{out}(N, M); P, \mathsf{in}^{\overline{o}}(N, x); Q\}\!\}, \mathcal{A}, \Lambda \xrightarrow{pre(\overline{o}, M)}_i \{\!\{P, Q\{^M/_x\}\}\!\}, \mathcal{A}, \Lambda \qquad \text{(I-C}\textsc{omm})$$

$$\{\!\{\mathsf{out}(N, M); P\}\!\}, \mathcal{A}, \Lambda \to_i \{\!\{P\}\!\}, \mathcal{A} \cup \{M\}, \Lambda \quad \text{if } N \in \mathcal{A} \qquad \text{(I-O}\textsc{ut})$$

$$\{\!\{\mathsf{in}^{\overline{o}}(N, x); Q\}\!\}, \mathcal{A}, \Lambda \xrightarrow{pre(\overline{o}, M)}_i \{\!\{Q\{^M/_x\}\}\!\}, \mathcal{A}, \Lambda \quad \text{if } N, M \in \mathcal{A} \qquad \text{(I-I}\textsc{n})$$

$$\{\!\{\mathsf{event}(M); P\}\!\}, \mathcal{A}, \Lambda \xrightarrow{M}_i \{\!\{P\}\!\}, \mathcal{A}, \Lambda \qquad \text{(I-E}\textsc{vent})$$

$$\emptyset, \mathcal{A}, \Lambda \to_i \emptyset, \mathcal{A} \cup \{M\}, \Lambda$$
$$\text{if } M_1, \ldots, M_n \in \mathcal{A},\ f/n \in \mathcal{F}_c \cup \mathcal{F}_d \text{ and } f(M_1, \ldots, M_n) \Downarrow M \qquad \text{(I-A}\textsc{pp})$$

$$\emptyset, \mathcal{A}, \Lambda \to_i \emptyset, \mathcal{A} \cup \{b_0[\lambda]\}, \Lambda$$
$$\text{with } b_0 \text{ a fixed name not appearing in } P \text{ or } Q, \text{ and } b_0[\lambda] \notin \mathcal{A},\ \lambda \text{ session identifier} \qquad \text{(I-G}\textsc{en})$$

$$\mathcal{P} \cup \mathcal{Q}, \mathcal{A}, \Lambda \xrightarrow{\alpha}_i \mathcal{P}' \cup \mathcal{Q}, \mathcal{A}', \Lambda'$$
$$\text{if } \mathcal{P}, \mathcal{A}, \Lambda \xrightarrow{\alpha}_i \mathcal{P}', \mathcal{A}', \Lambda' \qquad \text{(I-C}\textsc{ont})$$

Figure 3: Instrumented Semantics on Configurations

They formalise, intuitively, injectivity properties following from the freshness of occurrences, here $\overline{o}$. For example, one such axiom [CCT18] intuitively states that if a same trace T contains two events $ev = pre(\overline{o}, M)$ and $ev' = pre(\overline{o}, M')$, then $M = M'$.

We then define a notion of instrumented traces. It comes with a weaker variant that executes replicated inputs right after they are unfolded; this intuitively does not induce a loss of generality when constructing equivalence proofs, while giving more information to guide the analysis.

Definition 7 (Instrumented trace). *A sequence of transitions $\mathcal{C}_0 \xrightarrow{\ell_1}_i \ldots \xrightarrow{\ell_n}_i \mathcal{C}_n$ is called an* instrumented trace*. We also write $\mathcal{C} \xrightarrow{\ell}_{wi} \mathcal{C}'$ (*weak transition*) when:*

- *either $\mathcal{C} \xrightarrow{\ell}_i \mathcal{C}'$ using any rule except* I-Repl*, or* I-Repl *if the replicated process does not start with an input;*

- *otherwise $\ell = \ell_1 \cdot \ell_2$ and $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}'' \xrightarrow{\ell_2}_i \mathcal{C}'$ where $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}''$ is derived by rule* I-Repl *and $\mathcal{C}'' \xrightarrow{\ell_2}_i \mathcal{C}'$ is the application of the rule* I-In *or* I-Comm *using the input at the start of the process replicated in $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}''$.*

We write $\mathsf{wtrace}(\mathcal{C})$ *the set of $\to_{wi}$-traces of $\mathcal{C}$, and* $\mathsf{wrtrace}(\mathcal{C})$ *its more permissive variant where the last transition of the trace may be an arbitrary $\to_i$ transition.*

The connection between instrumented traces and regular traces is formalised in Lemma 6 in Appendix A.1.

On biprocesses In fact, to prove equivalence, ProVerif operates internally on *biprocesses*, that intuitively describe the joint execution of two processes to be proved equivalent. Their semantics is intuitively a straightforward extension of the instrumented semantics ensuring that the two paired processes follow the same execution flow. Formally:

Definition 8 (Biconfiguration)**.** *A* biconfiguration *is a tuple $\mathcal{C}^2 = \mathcal{P}^2, \mathcal{A}^2, \Lambda^2$ where $\mathcal{P}^2$ is a multiset of pairs of instrumented processes, $\mathcal{A}^2$ is a set of pairs of terms and Λ^2 is a set of pairs of pattern names. We say that $\mathcal{C}^2$ is* well-formed *when for all $(P, Q) \in \mathcal{P}^2$, we have $P \approx_{cf} Q$, and* initial *when it additionally verifies: $\Lambda^2 = \emptyset$, $\mathcal{A}^2 = \{(a[], a[]) \mid a \in \mathcal{N}_{pub}\}$ and all occurrence names in $\mathcal{P}^2$ appear at most once.*

The main difference with ProVerif's analogue is that our more permissive notion of control-flow equivalence $\approx_{cf}$, through the modelling of session decompositions as unordered multisets, makes the well-formedness condition much less restrictive. We define the projection functions:

$$proj_i(\mathcal{P}^2) = \{\!\{P_i \mid (P_1, P_2) \in \mathcal{P}^2\}\!\}$$

and, analogously, $proj_i(\mathcal{A}^2)$, $proj_i(\Lambda^2)$, $proj_i(\mathcal{C}^2)$. The semantics is then given by a transition relation $\xrightarrow{\ell}_{i^2}$, where ℓ is a pair of (possibly empty) events of the instrumented semantics, resulting in *bitraces*. The semantics is formalised in Figure 4

Moreover, we will need later on to distinguish replications followed by an input and replications followed by any other processes. As such, we split the set $\mathcal{N}_!$ into two distinct infinite sets $\mathcal{N}_{!i}$ and $\mathcal{N}_{!o}$ and request that in a (bi)configuration $\mathcal{C}$, any instance of a replication $!^{\overline{o}}_{\tilde{\mathsf{a}}}P$ should ensure that $o \in \mathcal{N}_{!i}$ if P starts with an input and $o \in \mathcal{N}_{!o}$ otherwise.

4.4 Convergence equivalence

Finally, similarly to the analogue notion of ProVerif for proving diff-equivalence [Bla09], we introduce an notion of *convergence* cast to our more general setting.

Definition 9 (Convergence)**.** *We say that a biconfiguration $\mathcal{C} = \mathcal{P}, \mathcal{A}, \Lambda$* converges*, denoted $\mathcal{C}{\downarrow}{\uparrow}$, when*

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

  $$N = L \textit{ iff } N' = L'$$

  Any communication that can be done by the left process can also be done by the right one, and conversely.

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

  $$f(M_1, \ldots, M_n) \Downarrow \mathsf{fail} \textit{ iff } f(M_1', \ldots, M_n') \Downarrow \mathsf{fail}$$

  The attacker does not observe any difference between the left and right sides. This is similar to static equivalence as defined in [ABF17].

$$\{\!\!\{(0,0)\}\!\!\}, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \emptyset, \mathcal{A}^2, \Lambda^2 \tag{I^2-Nil}$$

$$\{\!\!\{(P \mid Q, P' \mid Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \{\!\!\{(P, P'), (Q, Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \tag{I^2-Par}$$

$$\begin{array}{l} \{\!\!\{(\{\!\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}} P\}\!\!\} \cup \mathcal{M}, \{\!\!\{!^{\overline{o'}}_{\tilde{\mathsf{a}}'} P'\}\!\!\} \cup \mathcal{M}')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \xrightarrow{(repl(\overline{o}\sigma), repl(\overline{o'}\sigma'))}_{i^2} \\ \quad \{\!\!\{(P\sigma, P'\sigma'), (\{\!\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}} P\}\!\!\} \cup \mathcal{M}, \{\!\!\{!^{\overline{o'}}_{\tilde{\mathsf{a}}'} P'\}\!\!\} \cup \mathcal{M}')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \cup \{(\overline{o}\sigma, \overline{o'}\sigma')\} \\ \quad \text{if } dom(\sigma) = \tilde{\mathsf{a}},\ dom(\sigma') = \tilde{\mathsf{a}}',\ img(\sigma) \cup img(\sigma') \subseteq \mathbb{N}, \\ \quad \overline{o}\sigma \notin proj_0(\Lambda^2),\ \overline{o'}\sigma' \notin proj_1(\Lambda^2) \end{array} \tag{I^2-Repl}$$

$$\begin{array}{l} \{\!\!\{(\mathsf{out}(N, M); P, \mathsf{out}(N', M'); P'), (\mathsf{in}^{\overline{o}}(N, x); Q, \mathsf{in}^{\overline{o'}}(N', x'); Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \\ \quad \xrightarrow{(pre(\overline{o}, M), pre(\overline{o'}, M'))}_{i^2} \{\!\!\{(P, P'), (Q\{^M/_x\}, Q'\{^{M'}/_{x'}\})\}\!\!\}, \mathcal{A}^2, \Lambda^2 \end{array} \tag{I^2-Comm}$$

$$\begin{array}{l} \{\!\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \{\!\!\{(R, R')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \\ \quad \text{with } (R, R') = (P\{^M/_x\}, P'\{^{M'}/_{x'}\}) \text{ if } D \Downarrow M \\ \quad \text{and } D' \Downarrow M', \text{ and } (R, R') = (Q, Q') \text{ if } D \Downarrow \mathsf{fail} \text{ and } D' \Downarrow \mathsf{fail} \end{array} \tag{I^2-Let}$$

$$\begin{array}{l} \{\!\!\{(\mathsf{out}(N, M); P, \mathsf{out}(N', M'); P')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \{\!\!\{(P, P')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \cup \{(M, M')\} \\ \quad \text{if } (N, N') \in \mathcal{A}^2 \end{array} \tag{I^2-Out}$$

$$\begin{array}{l} \{\!\!\{(\mathsf{in}^{\overline{o}}(N, x); Q, \mathsf{in}^{\overline{o'}}(N', x'); Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \xrightarrow{(pre(\overline{o}, M), pre(\overline{o'}, M'))}_{i^2} \\ \quad \{\!\!\{(Q\{^M/_x\}, Q'\{^{M'}/_{x'}\})\}\!\!\}, \mathcal{A}^2, \Lambda^2 \\ \quad \text{if } (N, N'), (M, M') \in \mathcal{A}^2 \end{array} \tag{I^2-In}$$

$$\{\!\!\{(\mathsf{event}(M); Q, \mathsf{event}(M'); Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \xrightarrow{(M, M')}_{i^2} \{\!\!\{(Q, Q')\}\!\!\}, \mathcal{A}^2, \Lambda^2 \tag{I^2-Event}$$

$$\begin{array}{l} \emptyset, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \emptyset, \mathcal{A}^2 \cup \{(M, M')\}, \Lambda^2 \\ \quad \text{if } (M_1, M_1'), \ldots, (M_n, M_n') \in \mathcal{A},\ f/n \in \mathcal{F}_c \cup \mathcal{F}_d \text{ and } f(M_1, \ldots, M_n) \Downarrow M \\ \quad \text{and } f(M_1', \ldots, M_n') \Downarrow M' \end{array} \tag{I^2-App}$$

$$\emptyset, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \emptyset, \mathcal{A}^2 \cup \{(b_0[\lambda], b_0[\lambda])\}, \Lambda^2 \qquad \text{if } (b_0[\lambda], b_0[\lambda]) \notin \mathcal{A}^2 \tag{I^2-Gen}$$

$$\mathcal{P}^2 \cup \mathcal{Q}^2, \mathcal{A}^2, \Lambda^2 \xrightarrow{\alpha}_{i^2} \mathcal{P}^{2\prime} \cup \mathcal{Q}^2, \mathcal{A}^{2\prime}, \Lambda^{2\prime} \qquad \text{if } \mathcal{P}^2, \mathcal{A}^2, \Lambda^2 \rightarrow_{i^2} \mathcal{P}^{2\prime}, \mathcal{A}^{2\prime}, \Lambda^{2\prime} \tag{I^2-Cont}$$

Figure 4: Instrumented Semantics on (Well-Formed) Biconfigurations

- *if* $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q')\}\!\!\}$ *then*

  $$D \Downarrow \mathsf{fail} \textit{ iff } D' \Downarrow \mathsf{fail}$$

  The left process cannot take the "else" branch if the right process takes the "then" branch, and conversely.

By extension, we say that a bitrace is convergent if all of its intermediary biconfigurations are convergent.

The convergence property on a biconfiguration guarantees that any action possible on the left process can be applied on the right process, and vice-versa. This is formalised in the following lemma, that can be obtained through a straightforward inspection of the rules of Figures 3 and 4 under the assumption of convergence.

Lemma 1. *Let $\mathcal{C}$ be a well formed convergent biconfiguration and $i \in \{0, 1\}$. If $proj_i(\mathcal{C}) \xrightarrow{\ell_i}_i \mathcal{C}_i'$, then there exist ℓ_{1-i} and a well-formed biconfiguration $\mathcal{C}'$ such that $\mathcal{C} \xrightarrow{(\ell_0, \ell_1)}_{i^2} \mathcal{C}'$ and $proj_i(\mathcal{C}') = \mathcal{C}_i'$.*

Proof. Direct from Definition 9 and the semantics in Figures 3 and 4. □

As we are now focusing on bitraces and biconfigurations, we define "equivalences" based on convergent bitraces by considering a predicate π on configurations such that if C is the initial biconfiguration $(\{\!\{(P,Q)\}\!\}, \mathcal{A}, \emptyset)$ then $\pi(\mathcal{C})$ would imply $P \approx Q$. For example, a natural definition to define the predicate for observational equivalence would require that $\pi(\mathcal{C})$ implies that $\mathcal{C}$ is convergent and that if $\mathit{proj}_0(\mathcal{C}) \rightarrow_i \mathcal{C}'_1$ then $\mathcal{C} \rightarrow_{i^2} \mathcal{C}'$, $\mathit{proj}_0(\mathcal{C}') = \mathcal{C}'_1$ and $\pi(\mathcal{C}')$ for some $\mathcal{C}'$ (and similarly with $\mathit{proj}_1(\mathcal{C})$).

However, in Definition 3, we consider the weak equivalence, i.e. in property 2, one transition can be matched with any number of transitions. When working with biconfigurations and the transition relation $\rightarrow_{i^2}$, we force a strong observational equivalence, which can be too strong specifically when applying the rule I-Repl and I^2-Repl as the choice of how to match a process may depend on the first action of the replicated process.

Therefore, to define our predicates, we consider the weak transition relations $\xrightarrow{\ell}_{wi}$ and $\xrightarrow{\ell}_{wi^2}$ where $\mathcal{C} \xrightarrow{\ell}_{wi} \mathcal{C}'$ holds when, intuitively, all replications of processes P starting with an input must execute this input right away after a copy of P is unfolded. Formally:

- $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}_1 \xrightarrow{\ell_2}_i \mathcal{C}'$ with $\ell = \ell_1 \cdot \ell_2$, $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}_1$ is the application of the rule I-Repl and $\mathcal{C}_1 \xrightarrow{\ell_2}_i \mathcal{C}'$ is:
  1. either the application of the rule I-In on the process replicated in $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}_1$;
  2. either the rule I-Comm where the input corresponds to the process replicated in $\mathcal{C} \xrightarrow{\ell_1}_i \mathcal{C}_1$;
  3. an empty step (i.e., $\mathcal{C}' = \mathcal{C}_1$ and $\ell_2 = \epsilon$) if the replicated process does not start with an input.
- $\mathcal{C} \xrightarrow{\ell}_i \mathcal{C}'$ for all rules besides I-Repl.

We apply a similar definition for $\xrightarrow{\ell}_{wi^2}$.

We define $\mathsf{wrtrace}(\mathcal{C}_0)$ as the set of traces $T = \mathcal{C}_0 \xrightarrow{\ell_1}_i \mathcal{C}_1 \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$ such that for all $i \in \mathbb{N}$, if $\ell_i = \mathit{repl}(\bar{o})$ and $o \in \mathcal{N}_{!i}$ then $\mathcal{C}_{i-1} \xrightarrow{\ell_i \cdot \ell_{i+1}}_{wi} \mathcal{C}_{i+1}$. Additionally, we define $\mathsf{wtrace}(\mathcal{C}_0)$ as the set of traces $T = \mathcal{C}_0 \xrightarrow{\ell_1}_i \mathcal{C}_1 \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$ in $\mathsf{wrtrace}(\mathcal{C}_0)$ such that if $\ell_n = \mathit{repl}(\bar{o})$ then $o \notin \mathcal{N}_{!i}$. By definition, we have:

$$\mathsf{wtrace}(\mathcal{C}_0) \subset \mathsf{wrtrace}(\mathcal{C}_0) \subset \mathsf{trace}(\mathcal{C}_0)$$

Typically, the traces of $\mathsf{wtrace}(\mathcal{C}_0)$ are built by only applying the transition relation $\xrightarrow{\ell}_{wi}$. The traces $\mathsf{wrtrace}(\mathcal{C}_0)$ additionally allow the final transition step to be a replication that unfold a process starting with an input (which would not be possible by using $\xrightarrow{\ell}_{wi}$ as the input would need to be executed too).

Given a biconfiguration $\mathcal{C}$, we define similarly the sets $\mathsf{wtrace}^2(\mathcal{C})$ and $\mathsf{wrtrace}^2(\mathcal{C})$. We can now define new predicates based on convergent traces that are sound for the equivalences in Section 3.3.

Definition 10 (May-testing convergence)**.** *We say that a well formed biconfiguration $\mathcal{C}$ is* may-testing convergent, *denoted $\pi_{\approx_m}(\mathcal{C})$, when for all $i \in \{0,1\}$ and for all $T \in \mathsf{wtrace}(\mathit{proj}_i(\mathcal{C}))$, there exists a convergent bitrace $T' \in \mathsf{wtrace}^2(\mathcal{C})$ such that $\mathit{proj}_{1-i}(T') = T$. We say that $\mathcal{C}$ is pre-order may-testing convergent, denoted $\pi_{\sqsubseteq_m}(\mathcal{C})$, when the property is only required to hold for $i = 0$.*

Definition 11. Observational convergence equivalence *on well formed biconfigurations, denoted* $\pi_{\approx_o}$*, is defined as the largest predicate* π *such that* $\pi(\mathcal{C})$ *implies that* $\mathcal{C}$ *converges and, for all* $i \in \{0,1\}$*, if* $proj_i(\mathcal{C}) \xrightarrow{\ell_i}_{wi} \mathcal{C}'_i$ *then there exists a transition bi-step of the form* $\mathcal{C} \xrightarrow{(\ell_0,\ell_1)}_{wi^2} \mathcal{C}'$*, with* $proj_i(\mathcal{C}') = \mathcal{C}'_i$ *and* $\pi(\mathcal{C}')$*. We say that* $\mathcal{C}$ *is* pre-order convergent*, denoted* $\pi_{\sqsubseteq_o}(\mathcal{C})$ *when the predicate* $\pi(\mathcal{C}')$ *is only required to hold for* $i = 0$*.*

The soundness of Definitions 10 and 11 is given in our first main result.

Theorem 1 (Equivalence and convergence)**.** *Let* P, Q *be two processes. We also let* $\mathcal{A} = \{(a[], a[]) \mid a \in \mathcal{N}_{pub}\}$*, and the biconfiguration* $\mathcal{C} = (\{\!\{(\llbracket P \rrbracket_i, \llbracket Q \rrbracket_i)\}\!\}, \mathcal{A}, \emptyset)$*. If* $\mathcal{C}$ *is well formed then for all relations* $\mathcal{R} \in \{\approx_o, \sqsubseteq_o, \approx_m, \sqsubseteq_m\}$*,* $\pi_{\mathcal{R}}(\mathcal{C})$ *implies* $P \mathrel{\mathcal{R}} Q$*.*

A detailed proof of this result can be found in Appendix A.

4.5 Axioms

ProVerif recently introduced the possibilities to declare axioms to help it prove diff equivalence [BCC22]. Typically, axioms are properties that ProVerif will assume to hold on all bitraces and apply them when saturating the set of Horn clauses. Of course, to ensure the soundness, one needs to prove on the side that the declared axioms actually holds for any bitraces of the protocol. In this section, we prove some properties on the events labeling the transition relations of the semantics that will be considered as axioms by ProVerif.

Our properties focuses on traces and bitraces for the weak transition relations $\xrightarrow{\ell}_{wi}$ and $\xrightarrow{\ell}_{wi^2}$. As such, we consider a new event functions $repl_i$ in $\mathcal{F}_e$. Typically, when the transition $\mathcal{C} \xrightarrow{repl(\overline{o})\cdot pre(\overline{o'},M)}_{wi} \mathcal{C}'$ occurs, corresponding to a replication followed by an input, the event $repl_i$ will record the replication occurrence $\overline{o}$ as well as the input term M, i.e. $repl_i(\overline{o}, M)$. We say that the event function $repl_i$ is the *input* event function.

Definition 12 (Event satisfaction)**.** *Let* $\mathcal{C}$ *be a configuration. Let* $T \in \mathsf{wrtrace}(\mathcal{C})$ *be a trace* $\mathcal{C}_0 \xrightarrow{\ell_1}_i \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$ *and let* $i \in \mathbb{N}$*. We define the* satisfaction relation $\vdash$ of an event ev on T at step i*, denoted* $T, i \vdash ev$*, that holds when either* $ev = \ell_i$*; or* $ev = repl_i(\overline{o}, M)$ *and* $\mathcal{C}_{i-2} \xrightarrow{repl(\overline{o})\cdot pre(\overline{o}',M)}_{wi} \mathcal{C}_i$*.*

Given $\mathcal{C}$ *be a biconfiguration, a bitrace* $T \in \mathsf{wrtrace}^2(\mathcal{C})$ *and* $i \in \mathbb{N}$*, we define the* satisfaction relation on bievents (ev, ev')*, denoted* $T, i \vdash^2 (ev, ev')$*, that holds when* $proj_0(T), i \vdash ev$ *and* $proj_1(T), i \vdash ev'$*.*

We may write $T \vdash ev$ and $T \vdash^2 (ev, ev')$ when the step is unnecessary. Formally, they correspond to $\exists i \in \mathbb{N}. T, i \vdash ev$ and $\exists i \in \mathbb{N}. T, i \vdash^2 (ev, ev')$.

Notice that the event $repl_i(\overline{o}, M)$ can only be satisfied it corresponds to a weak transition $\xrightarrow{\ell}_{wi}$. Moreover, when $T, i \vdash repl_i(\overline{o}, M)$ then $T, i-1 \vdash repl(\overline{o})$. In particular, $repl(\overline{o})$ and $repl_i(\overline{o}, M)$ are called *replication events*, while the former is more specifically called a *strict replication event.* We denote by $\mathsf{Ev}_!$ (resp. $\mathsf{Ev}_{!i}$) the set of strict replication events (resp. of replication events of the form $repl_i(\overline{o}, M)$). Moreover, we say that $\overline{o}$ is the *replication occurrence of* ev, or *replication name pattern*, denoted $\mathsf{orepl}(ev)$.

We can now state several properties in the following lemma (proof can be found in Appendix B).

Lemma 2. *Let $\mathcal{C}$ be an initial instrumented configuration. For all $i, j \in \mathbb{N}$, for all $T \in \mathsf{wrtrace}(\mathcal{C})$,*

- *if $T, i \vdash pre(\overline{o_1}, M)$ and $T, j \vdash pre(\overline{o_2}, N)$; or $T, i \vdash repl_i(\overline{o_1}, M)$ and $T, j \vdash repl_i(\overline{o_2}, N)$ then*

  1. *$\overline{o_1} = \overline{o_2}$ if and only if $i = j$*
  2. *$i = j$ implies $M = N$*

- *if $T, i \vdash ev$ and $T, j \vdash ev'$ with $ev, ev' \in \mathsf{Ev}_!$ then*

  1. *$\mathsf{orepl}(ev) = \mathsf{orepl}(ev')$ implies $i = j$*
  2. *$i = j$ implies $ev = ev'$*

- *if $T \vdash ev$ and $T \vdash ev'$ with $ev, ev' \in \mathsf{Ev}_!$, $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$ and $\mathsf{orepl}(ev) = o[\tilde{\mathsf{b}}]$ then $\tilde{\mathsf{a}}_{|\lambda} = \tilde{\mathsf{b}}_{|\lambda}$ implies $\tilde{\mathsf{a}} = \tilde{\mathsf{b}}$.*

Relying on Lemmas 2, we can define similar properties on bitraces that focus separately on the left and right sides of events. We can also define properties that link the left and right sides of replication events. In particular, replication names on the left side of the biconfiguration cannot be matched with any replication names on the right side. Given an initial biconfiguration, we can compute the set of replication names on the right (resp. left) side a replication name on the left (resp. right) side should be matched with. This is formalized as follows. Given a process P with distinct occurrence names, we denote by $names_!(P)$ the set of occurrence names occurring in a matching composition, i.e. $o \in names_!(P)$ when $!^{\overline{o}}_{\tilde{\mathsf{a}}} P'$ occurs in P. We augment this notation to initial instrumented configurations $\mathcal{C}$, denoted $names_!(\mathcal{C})$.

Definition 13 (Potential matching)**.** *Let P, Q two processes such that $P \approx_{cf} Q$. We define the set $\mathsf{pm}(P, Q)$, called the* set of potential matching replication occurrence names in P and Q, *as the set of pairs of replication occurrence names such that:*

$$\mathsf{pm}(0, 0) = \emptyset$$
$$\mathsf{pm}(\mathsf{out}(N, M); P, \mathsf{out}(N', M'); P') = \mathsf{pm}(P, P')$$
$$\mathsf{pm}(\mathsf{in}^{\overline{o}_1}(N, x); P, \mathsf{in}^{\overline{o}'_1}(N', x'); P') = \mathsf{pm}(P, P')$$
$$\mathsf{pm}(\mathcal{M}, \mathcal{M}') = \{(o, o') \mid !^{\overline{o}}_{\tilde{\mathsf{a}}} P \in \mathcal{M} \wedge !^{\overline{o}'}_{\tilde{\mathsf{a}}'} P' \in \mathcal{M}'\} \cup \bigcup_{!^{\overline{o}}_{\tilde{\mathsf{a}}} P \in \mathcal{M}} \bigcup_{!^{\overline{o}'}_{\tilde{\mathsf{a}}'} P' \in \mathcal{M}'} \mathsf{pm}(P, P')$$
$$\mathsf{pm}(\mathsf{event}(ev); P, \mathsf{event}(ev'); P') = \mathsf{pm}(P, P')$$
$$\mathsf{pm}(P_1 \mid P_2, P'_1 \mid P'_2) = \mathsf{pm}(P_1, P'_1) \cup \mathsf{pm}(P_2, P'_2)$$
$$\mathsf{pm}(\mathsf{let}\ x = D\ \mathsf{in}\ P_1\ \mathsf{else}\ P_2, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'_1\ \mathsf{else}\ P'_2) = \mathsf{pm}(P_1, P'_1) \cup \mathsf{pm}(P_2, P'_2)$$

Given an initial instrumented biconfiguration $\mathcal{C} = (\{\!\{(P, Q)\}\!\}, \mathcal{A}, \emptyset)$, we denote by $\mathrm{pm}(C)$ *the set* $\mathsf{pm}(P, Q)$.

We also sometimes need to refer to the arguments of an occurrence replication names in (bi)configuration. Hence, given $o \in \mathcal{N}_!$, we denote by $\mathsf{ar}_{\mathcal{C}}(o)$, $\mathsf{ar}^{\lambda}_{\mathcal{C}}(o)$, $\mathsf{ar}^{b}_{\mathcal{C}}(o)$ respectively the numbers $|\tilde{\mathsf{a}}|$, $|\tilde{\mathsf{a}}_{|\lambda}|$ and $|\tilde{\mathsf{b}}|$ when $!^{o[\tilde{\mathsf{a}}]}_{\tilde{\mathsf{b}}} P$ occurs in $\mathcal{C}$ (recall the all occurrence names are distinct in the initial instrumented configuration so there can be only one possible value).

Example 2. Coming back to our running example, we have $\mathrm{pm}(\mathcal{C}_{\mathsf{BAC}}) = \{(o_3, o_1); (o_4, o_2)\}$, $\mathsf{ar}^{\lambda}_{\mathcal{C}}(o_1) = \mathsf{ar}^{\lambda}_{\mathcal{C}}(o_2) = 1$ and $\mathsf{ar}^{\lambda}_{\mathcal{C}}(o_3) = \mathsf{ar}^{\lambda}_{\mathcal{C}}(o_4) = 2$.

Given an initial instrumented biconfiguration $\mathcal{C}$, we also define a partial order relation between replication names, denoted $o \prec_{\mathcal{C}} o'$ when o' is in the scope of o in the processes of $\mathcal{C}$. We denote by $o \preceq_{\mathcal{C}} o'$ when $o \prec_{\mathcal{C}} o'$ or $o = o'$. We extend this notation to replication name patterns such that $o[\tilde{\mathsf{a}}] \prec_{\mathcal{C}} o'[\tilde{\mathsf{a}}']$ when $o \prec_{\mathcal{C}} o'$ and $\tilde{\mathsf{a}}_{|\lambda}$ is a prefix of $\tilde{\mathsf{a}}'_{|\lambda}$. Finally, we sometimes assimilate $o[\tilde{\mathsf{a}}]$ with the name o and write $(o[\tilde{a}], o'[\tilde{\mathsf{a}}']) \in \mathrm{pm}(\mathcal{C})$ instead of $(o, o') \in \mathrm{pm}(\mathcal{C})$.

Lemma 3 (Consistency)**.** *Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in$ $\mathsf{wrtrace}^2(\mathcal{C})$, if $T, i \vdash^2 (ev_0, ev_1)$ and $T, j \vdash^2 (ev'_0, ev'_1)$ with $ev_0, ev_1, ev'_0, ev'_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ then*

- $(\mathsf{orepl}(ev_0), \mathsf{orepl}(ev_1)) \in \mathrm{pm}(\mathcal{C})$
- $\mathsf{orepl}(ev_0) \prec_{\mathcal{C}} \mathsf{orepl}(ev'_0)$ *if and only if* $\mathsf{orepl}(ev_1) \prec_{\mathcal{C}} \mathsf{orepl}(ev'_1)$

The proof of this lemma can be found in Appendix B. Intuitively, Lemma 3 states that all labels of the semantics rules always contain matching replication names with one another and preserve the scope of replication names.

Using the relation $\prec_{\mathcal{C}}$, we can also extend Lemma 1 to indicate which replication name patterns can selected when applying the transition rule I^2-Repl (the proof of the following lemma can be found in Appendix B).

Lemma 4. *Let $\mathcal{C}_0$ be an initial convergent instrumented biconfiguration. Let $\mathcal{C}_0 \overset{\mathsf{tr}}{\Rightarrow}_{i^2} \mathcal{C}_1 =$ $\mathcal{P}, \mathcal{A}, \Lambda$. Let $i \in \{0,1\}$.*

Assume $proj_i(\mathcal{C}_1) \xrightarrow{repl(o_i[\tilde{\mathsf{a}}_i])}_i \mathcal{C}'_2$. For all o_{1-i} such that $k = \mathsf{ar}^{\lambda}_{\mathcal{C}_0}(o_{1-i})$, for all $\lambda_1, \dots, \lambda_k \in$ $\mathbb{N}$, if $(o_0, o_1) \in \mathrm{pm}(\mathcal{C}_0)$ and for all $(o'_0[\tilde{\mathsf{a}}'_0], o'_1[\tilde{\mathsf{a}}'_1]) \in \Lambda$,

- *$o'_i[\tilde{\mathsf{a}}'_i] \prec_{\mathcal{C}_0} \overline{o}_i$ implies $o'_{1-i}[\tilde{\mathsf{a}}'_{1-i}] \prec_{\mathcal{C}_0} o_{1-i}[\lambda_1, \dots, \lambda_k]$*
- *$o'_{1-i}[\tilde{\mathsf{a}}'_{1-i|\lambda}] \neq o_{1-i}[\lambda_1, \dots, \lambda_k]$*

then $\mathcal{C}_1 \xrightarrow{(repl(o_0[\tilde{\mathsf{a}}_0]), repl(o_1[\tilde{\mathsf{a}}_1]))}_{i^2} \mathcal{C}_2$ for some $\tilde{\mathsf{a}}_{1-i}$ with $proj_i(\mathcal{C}_2) = \mathcal{C}'_2$ and $\tilde{\mathsf{a}}_{1-i|\lambda} = \lambda_1, \dots, \lambda_k$.

5 Clause generation

5.1 Clauses for the attacker

Below we display the clauses modelling the capabilities of the attacker. They are adapted from [BCC22]. We use the notations for clauses (ev, att...) as in Section 2. We consider the set of public names $\mathcal{A}_0$ in the initial instrumented biconfiguration $\mathcal{C} = (\{\!\{(P, Q)\}\!\}, \mathcal{A}_0, \emptyset)$.

$$
\begin{array}{lr}
\text{For each } a \in \mathcal{A}_0, \mathrm{att}(a[], a[]) & \text{(RInit)} \\
\mathrm{att}(b_0[i], b_0[i]) & \text{(RGen)} \\
\mathrm{att}(\mathsf{fail}, \mathsf{fail}) & \text{(RFail)} \\
\text{For each function } h, \text{ for all } h(U_1, \dots, U_m) \to U \mid\mid \phi \text{ in } \mathsf{def}(h), & \\
\quad \text{for all } h(U'_1, \dots, U'_m) \to U' \mid\mid \phi' \text{ in } \mathsf{def}(h), & \text{(Rf)} \\
\quad \mathrm{att}(U_1, U'_1) \wedge \dots \wedge \mathrm{att}(U_m, U'_m) \wedge \phi \wedge \phi' \to \mathrm{att}(U, U') & \\
\mathrm{msg}(x, y, x', y') \wedge \mathrm{att}(x, x') \to \mathrm{att}(y, y') & \text{(Rl)}
\end{array}
$$

$$\mathrm{att}(x,x') \wedge \mathrm{att}(y,y') \rightarrow \mathrm{msg}(x,y,x',y') \tag{Rs}$$

$$\mathrm{att}(x,x') \rightarrow \mathrm{input}(x,x') \tag{RIn}$$

$$\mathrm{input}(x,y) \wedge \mathrm{msg}(x,z,y',z') \wedge y \neq y' \rightarrow \mathrm{bad} \tag{RIBad1}$$

$$\mathrm{input}(y,x) \wedge \mathrm{msg}(y',z,x,z') \wedge y \neq y' \rightarrow \mathrm{bad} \tag{RIBad2}$$

$$\mathrm{att}(x,\mathsf{fail}) \rightarrow \mathrm{bad} \tag{RBad1}$$

$$\mathrm{att}(\mathsf{fail},x) \rightarrow \mathrm{bad} \tag{RBad2}$$

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

5.2 Clauses for the protocol

The clauses modelling the protocols P and Q are generated by the translation $\llbracket|P,Q|\rrbracket\mathcal{H}r$, displayed below, where $\mathcal{H}$ is a conjunction of facts and formulas and r is either a term or $\square$.

Intuitively, $\mathcal{H}$ collects conditions that need to be satisfied for P and Q to be executed, and $r = \mathsf{diff}[o_1,o_2]$ indicates that P and Q have been spawned with occurrence replications o_1 and o_2 respectively. In particular, it should be recorded in $\mathcal{H}$ that o_1 and o_2 should be matched together during an equivalence proof. This is done through the facts $F_{!_i}(r,x,x')$ and $F_!(r)$ whose values are $\top$ when $r = \square$ and otherwise when $r = \mathsf{diff}[o_1,o_2]$:

$$F_{!_i}(r,x,x') = \mathrm{ev}(\mathit{repl}_i(o_1,x), \mathit{repl}_i(o_2,x'))$$
$$F_!(r) = \mathrm{ev}(\mathit{repl}(o_1), \mathit{repl}(o_2))$$

Note that we consider that the variables in processes P and Q are bound only once. We freshly renamed them if it is not the case before generating the clauses.

$$\llbracket|0,0|\rrbracket\mathcal{H}r = \emptyset$$

$$\llbracket|\mathcal{M},\mathcal{M}'|\rrbracket\mathcal{H}r = \bigcup_{!^{\overline{o_1}}_{\tilde{\mathsf{a}}_1}P_1\in\mathcal{M}} \; \bigcup_{!^{\overline{o_2}}_{\tilde{\mathsf{a}}_2}P_2\in\mathcal{M}'} \llbracket|P_1,P_2|\rrbracket\mathcal{H}\mathsf{diff}[\overline{o_1},\overline{o_2}]$$

$$\llbracket|P \mid Q, P' \mid Q'|\rrbracket\mathcal{H}r = \llbracket|P,P'|\rrbracket\mathcal{H}\square \cup \llbracket|Q,Q'|\rrbracket\mathcal{H}\square$$

$$\llbracket|\mathsf{in}^{\overline{o}}(N,x);P,\mathsf{in}^{\overline{o'}}(N',x');P'|\rrbracket\mathcal{H}r =$$
$$\quad \llbracket|P,P'|\rrbracket(\mathcal{H} \wedge \mathrm{msg}(N,x,N',x') \wedge \mathrm{ev}(\mathit{pre}(\overline{o},x),\mathit{pre}(\overline{o'},x')) \wedge F_{!_i}(r,x,x'))\square$$
$$\quad \cup \{\mathcal{H} \wedge F_!(r) \rightarrow \mathrm{input}(N,N')\}$$

$$\llbracket|\mathsf{out}(N,M);P,\mathsf{out}(N',M');P'|\rrbracket\mathcal{H}r = \llbracket|P,P'|\rrbracket(\mathcal{H} \wedge F)\square \cup \{\mathcal{H} \wedge F_!(r) \rightarrow \mathrm{msg}(N,M,N',M')\}$$

$$\llbracket|\mathsf{event}(ev);P,\mathsf{event}(ev');P'|\rrbracket\mathcal{H}r = \llbracket|P,P'|\rrbracket(\mathcal{H} \wedge F_!(r) \wedge \mathrm{ev}(ev,ev'))\square$$

$$\llbracket|\mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'\ \mathsf{else}\ Q'|\rrbracket\mathcal{H}r =$$
$$\quad \bigcup\{\llbracket|P\sigma\sigma',P'\sigma\sigma'|\rrbracket(\mathcal{H}\sigma \wedge F_!(r)\sigma \wedge \phi)\square \mid (D,D') \Downarrow' ((M,M'),\sigma,\phi) \wedge \sigma' = \{^{M}/_{x}, ^{M'}/_{x'}\}\}$$
$$\quad \cup \bigcup\{\llbracket|Q\sigma,Q'\sigma|\rrbracket(\mathcal{H}\sigma \wedge F_!(r)\sigma \wedge \phi)\square \mid (D,D') \Downarrow' ((\mathsf{fail},\mathsf{fail}),\sigma,\phi)\}$$
$$\quad \cup \{\mathcal{H}\sigma \wedge F_!(r)\sigma \wedge \phi \rightarrow \mathrm{bad} \mid (D,D') \Downarrow' ((\mathsf{fail},M),\sigma,\phi)\}$$
$$\quad \cup \{\mathcal{H}\sigma \wedge F_!(r)\sigma \wedge \phi \rightarrow \mathrm{bad} \mid (D,D') \Downarrow' ((M,\mathsf{fail}),\sigma,\phi)\}$$

Given an initial instrumented biconfiguration $\mathcal{C} = (\{\!\{(P,Q)\}\!\}, \mathcal{A}_0, \emptyset)$, we will denote by $\mathbb{C}_{\mathcal{P}}(\mathcal{C})$ the set of Horn clauses $\llbracket |P,Q| \rrbracket \top\square \cup \mathbb{C}_{\mathcal{A}}(\mathcal{C})$. Furthermore, given a bitrace T, we define the set of Horn clauses $\mathbb{C}_e(T) = \{\rightarrow \mathrm{ev}(ev, ev') \mid T, i \vdash^2 (ev, ev') \text{ and } i \in \mathbb{N}\}$.

Example 3. For our running example, we translate the instrumented processes $\{\!\{!^{o_3[i,j]}_{i,j} P_{\mathit{left}}\}\!\} \mid \{\!\{!^{o_4[i,j]}_{i,j} R_{\mathit{left}}\}\!\}$ and $\{\!\{!^{o_1[i]}_{i} P_{\mathit{right}}\}\!\} \mid \{\!\{!^{o_2[i]}_{i} R_{\mathit{right}}\}\!\}$. Let us look more closely at the passport components, i.e. the translation $\llbracket | \{\!\{!^{o_3[i,j]}_{i,j} P_{\mathit{left}}\}\!\}, \{\!\{!^{o_1[i]}_{i} P_{\mathit{right}}\}\!\} | \rrbracket \top\square$.

Recall that by instrumentation, P_{left} and P_{right} are the processes $P(k_\ell[i])$ and $P(k_r[i'])$ where the inputs have been associated with the occurrences $o'_\ell[i,j]$ and $o'_r[i']$, and the name n has been replaced by $n_\ell[i,j]$ and $n_r[i']$ respectively.

The translation $\llbracket | \{\!\{!^{o_3[i,j]}_{i,j} P_{\mathit{left}}\}\!\}, \{\!\{!^{o_1[i']}_{i'} P_{\mathit{right}}\}\!\} | \rrbracket \top\square$ will then record the replication occurrences $o_3[i,j]$ and $o_1[i']$. As the first actions in the processes P_{left} and P_{right} are outputs, it results:

$$F_! \rightarrow \mathrm{msg}(c, n_\ell[i,j], c, n_r[i'])$$

with $F_! = \mathrm{ev}(\mathit{repl}(o_3[i,j]), \mathit{repl}(o_1[i']))$ being propagated in the hypotheses of the remaining clauses. When translating the second actions, i.e., the inputs $\mathsf{in}^{o'_\ell[i,j]}(c,x)$ and $\mathsf{in}^{o'_r[i']}(c,x')$, only the precise event is added in the hypothesis and not the occurrence fact $F_{!_i}$ as such fact is only added when translating the first actions after a session matching. This yields the following hypothesis H to be propagated:

$$F_! \wedge \mathrm{msg}(c,x,c,x') \wedge \mathrm{ev}(\mathit{pre}(o'_\ell[i,j],x), \mathit{pre}(o'_r[i'],x'))$$

Finally, as in ProVerif, going through the conditional branch, we compute the success and failure conditions before translating the final output actions, yielding the following clauses:

$$\begin{array}{l}
H\sigma_1\sigma_2 \rightarrow \mathrm{msg}(c, \mathsf{ok}, c, \mathsf{ok}) \\
H\sigma_1 \wedge \forall z'.x' \neq \mathsf{senc}(z', k_r[i']) \rightarrow \mathrm{msg}(c, \mathsf{ok}, c, \mathsf{error}) \\
H\sigma_2 \wedge \forall z.x \neq \mathsf{senc}(z, k_\ell[i]) \rightarrow \mathrm{msg}(c, \mathsf{error}, c, \mathsf{ok}) \\
H \wedge \forall z'.x' \neq \mathsf{senc}(z', k_r[i']) \wedge \forall z.x \neq \mathsf{senc}(z, k_\ell[i,j]) \\
\qquad \rightarrow \mathrm{msg}(c, \mathsf{error}, c, \mathsf{error})
\end{array}$$

with $\sigma_1 = \{^{\mathsf{senc}(z,k_\ell[i])}/_{x}\}$ and $\sigma_2 = \{^{\mathsf{senc}(z',k_r[i'])}/_{x'}\}$.

As mentioned, we also consider the clauses $\mathbb{C}_{\mathcal{A}}(\mathcal{C}^2)$ describing the attacker capabilities. They include in particular:

$$\begin{array}{c}
\rightarrow \mathrm{att}(c,c) \qquad \rightarrow \mathrm{att}(\mathsf{ok},\mathsf{ok}) \qquad \rightarrow \mathrm{att}(\mathsf{error},\mathsf{error}) \\
\mathrm{msg}(x,y,x',y') \wedge \mathrm{att}(x,x') \rightarrow \mathrm{att}(y,y') \\
\mathrm{att}(x,y) \wedge \mathrm{att}(x,y') \wedge y \neq y' \rightarrow \mathrm{bad}
\end{array}$$

The first three clauses represents the fact that the attacker knows the initial constants. The fourth clause indicates that if the attacker knows a channel, it can learn the messages sent over it. Finally, the fifth clause models the fact that by deducing twice the same messages on the left side but different messages on the right side, the attacker can distinguish the two traces, i.e. the bitrace is not convergent. In our example, it translates the fact that a bitrace satisfying the conditional in the passport on one side side but failing it on the other side will lead to a non-convergent bitrace. Note that the fifth clause comes from the clauses Rf by application of the destructor Equals.

5.3 Soundness

In this work, we use the saturation procedure of the recent release of ProVerif that applies axioms, lemmas and restrictions when saturating the set of Horn clauses. In our framework, though we do not generate exactly the same set of Horn clauses (we in fact generate a more general set of clauses), we do not consider here user defined lemmas, axioms and restrictions thus we can simplify the definitions, theorems and proofs in [] to fit our needs.

First, let us recall the notions of *subsumption of Horn clauses* and *derivation of facts*

Definition 14. *Let $H_1 \wedge \phi_1 \rightarrow C_1$ and $H_2 \wedge \phi_2 \rightarrow C_2$ be two clauses. We say that $H_1 \wedge \phi_1 \rightarrow C_1$ subsumes $H_2 \wedge \phi_2 \rightarrow C_2$, denoted $(H_1 \wedge \phi_1 \rightarrow C_1) \sqsupseteq (H_2 \wedge \phi_2 \rightarrow C_2)$, 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 15. *Let $\mathbb{C}$ be a set of clauses. Let F be a closed fact and a step τ. A derivation $\mathcal{D}$ of F from $\mathbb{C}$ is a finite tree defined as follows:*

- *its nodes (except the root) are labelled by clauses $R \in \mathbb{C}$.*
- *its edges are labelled by ground facts.*
- *if the tree contains a node labelled by R with one incoming edge labelled by F_0 and n outgoing edges labelled 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, labelled by F.*

The following theorem states that the fact bad is derivation from the set of generated Horn clauses when there is a non convergent bitrace. Considering that we change the set of Horn clauses generated w.r.t. the ones generated in [BCC22], we prove this theorem in Appendix C. Note that its proof is heavily inspired by the proof of [BCC22, Lemma 11].

Theorem 2 (Soundness Initial Clauses)**.** *Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in \mathsf{wrtrace}^2(\mathcal{C})$, if T does not converge then there exists a derivation of* bad *from $\mathbb{C}_{\mathcal{P}}(\mathcal{C}) \cup \mathbb{C}_e(T)$.*

The saturation procedure, that we denote $\mathsf{saturate}(\mathbb{C})$, preserves derivation of bad as stated in the following theorem from [BCC22].

Proposition 1. *[[BCC22, Theorem 4]] Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in \mathsf{wrtrace}^2(\mathcal{C})$, if there exists a derivation of* bad *from $\mathbb{C}_{\mathcal{P}}(\mathcal{C}) \cup \mathbb{C}_e(T)$ then there exists a derivation of* bad *from $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C})) \cup \mathbb{C}_e(T)$.*

Corollary 1 (Soundness Saturation)**.** *Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in \mathsf{wrtrace}^2(\mathcal{C})$, if T does not converge then there exists a clause $H \wedge \phi \rightarrow \mathrm{bad}$ in $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}))$ and a substitution σ such that $\sigma \models \phi$ and for all $\mathrm{ev}(ev, ev') \in H$, there exists $i \in \mathbb{N}$ such that $T, i \vdash^2 (ev, ev')\sigma$.*

Proof. Using Proposition 1 and theorem 2, we obtain that if T does not converge then there exists a derivation of bad from $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C})) \cup \mathbb{C}_e(T)$. By definition of a derivation (Definition 15), we conclude by noticing that the root of the derivation is labelled by a clause concluding bad, i.e. $H \wedge \phi \to \text{bad}$. By definition of a derivation, we do have a substitution σ such that $\sigma \models \phi$. Moreover, since $\mathbb{C}_e(T)$ are the only clauses concluding events, we conclude by definition of $\mathbb{C}_e(T)$. □

Example 4. If $\mathcal{C}^2_{\mathsf{BAC}}$ is an initial biconfiguration corresponding to the processes of the running example, the set $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}^2_{\mathsf{BAC}}))$ contains two clauses deriving bad:

$$\begin{aligned} C_1 = {} & \text{ev}(\mathit{repl}(o_3[i_1, j_1]), \mathit{repl}(o_1[i])) \wedge \\ & \text{ev}(\mathit{repl}_i(o_4[i_2, j_2], n_l[i_1, j_1]), \mathit{repl}_i(o_2[i], n_r[i])) \wedge \\ & H^1_{\mathit{pre}} \wedge i_1 \neq i_2 \to \text{bad} \end{aligned}$$

$$\begin{aligned} C_2 = {} & \text{ev}(\mathit{repl}(o_3[i, j]), \mathit{repl}(o_1[i_1])) \wedge \\ & \text{ev}(\mathit{repl}_i(o_4[i, j'], n_l[i, j]), \mathit{repl}_i(o_2[i_2], n_r[i_1])) \wedge \\ & H^2_{\mathit{pre}} \wedge i_1 \neq i_2 \to \text{bad} \end{aligned}$$

The first clause corresponds to the case where on S_{right}, the nonce $n_r[i]$ of the reader was sent to the correct passport, as indicated by $\mathit{repl}_i(o_2[i], n_r[i])$, thus ok will be output. On the left side however, $\mathit{repl}_i(o_4[i_2, j_2], n_l[i_1, j_1])$ and $i_1 \neq i_2$ indicate that the nonce $n_l[i_1, j_1]$ generated by a reader having the key $k_l[i_1]$ was sent to a passport with the key $k_l[i_2]$. Since $i_1 \neq i_2$, the keys are different hence error will be output. The second clause is the dual of the first one, i.e., the test will succeed on S_{left} but fail on S_{right}. Here, H^1_{pre} and H^2_{pre} contains the precise events (omitted). The sets of initial and saturated clauses can be displayed by running our implementation [Ano23b] on the file running_example.pv

6 Semantics Conditions for Equivalence

In this section, we provide several tests on saturated clauses for proving the equivalence predicates and their pre-order defined in Section 4.4, that are $\pi_{\approx_o}$, $\pi_{\sqsubseteq_o}$, $\pi_{\approx_m}$ and $\pi_{\sqsubseteq_m}$.

6.1 Completion of Horn clauses

Looking at the initial generation of Horn clauses from a protocol, we have some structural properties on the events occurring in the clauses that are intuitively the direct translation of Lemmas 2 and 3. For instance, in a clause $H \to F$, if the fact $\text{ev}(\mathit{repl}(o_0[\tilde{\mathsf{a}}_0]), \mathit{repl}(o_1[\tilde{\mathsf{a}}_1]))$ occurs in H and there are $o'_0 \prec_{\mathcal{C}_I} o_0$ then there must exists in H a fact $\text{ev}(\mathit{repl}(o'_0[\tilde{\mathsf{a}}'_0]), \mathit{repl}(o'_1[\tilde{\mathsf{a}}'_1]))$ with $(o'_0, o'_1) \in \text{pm}(\mathcal{C}_I)$ and $o'_i[\tilde{\mathsf{a}}'_i] \prec_{\mathcal{C}_I} o_i[\tilde{\mathsf{a}}_i]$ for $i = 1, 2$.

In ProVerif, the saturation procedure may break this invariant by deleting some events (as deleting events is always sound). However, thanks to Lemmas 2 and 3, we can always reconstruct the missing events to retrieve the property stated above. Thus, in the rest of this technical report, we assume that such property holds.

6.2 Session matching

We now arrive to the main tool for matching sessions. Recall that to prove may-testing, we need to show that for all traces T in $\mathit{proj}_0(\mathcal{C}^2)$, we can find a corresponding convergent bitrace T^2 of C^2 with $proj_0(T^2) = T$. In particular, for every replication event satisfied by T, e.g. $T \vdash ev = repl(o[\tilde{\mathsf{a}}])$, there must be a corresponding $ev' = repl(o'[\tilde{\mathsf{a}}'])$ such that $T^2 \vdash^2 (ev, ev')$. Intuitively, a session matching is a function that associates each such event ev with a corresponding occurrence $o'[\tilde{\mathsf{a}}']$. However, $\tilde{\mathsf{a}}'$ may not only contain session identifiers but also terms corresponding to previous inputs. To avoid reasoning on the messages input in $proj_1(T^2)$, we strip $\tilde{\mathsf{a}}'$ from these input terms, i.e. $\tilde{\mathsf{a}}'_{|\lambda}$, to only preserve the session identifiers. In such a case, we say that $o'[\tilde{\mathsf{a}}'_{|\lambda}]$ is a *pure replication pattern*. We denote by $\mathsf{P}_!$ the set of ground pure replication pattern.

Definition 16 (Session matching). *Let $\mathcal{C} = (\{\!\{(P, Q)\}\!\}, \mathcal{A}, \emptyset)$ be an initial instrumented bi-configuration. We say that a partial mapping ρ from ground events in $\mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ to $\mathsf{P}_!$ is a* session matching from P to Q *when for all $ev, ev_1, ev_2 \in dom(\rho)$, if $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$, $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ and $\mathsf{orepl}(ev_2) = o_2[\tilde{\mathsf{a}}_2]$ then*

- *$\rho(ev) = o'[\tilde{\mathsf{a}}']$ implies $o \in names_!(P)$, $(o, o') \in \mathrm{pm}(\mathcal{C})$ and $\mathsf{ar}^\lambda_{\mathcal{C}}(o') = |\tilde{\mathsf{a}}'|$;*
- *$\mathsf{orepl}(ev_1) \prec_{\mathcal{C}} \mathsf{orepl}(ev_2)$ if and only if $\rho(ev_1) \prec_{\mathcal{C}} \rho(ev_2)$.*
- *$ev_1 \in \mathsf{Ev}_!$, $ev_2 \in \mathsf{Ev}_{!i}$ and $\mathsf{orepl}(ev_1) = \mathsf{orepl}(ev_2)$ implies $\rho(ev_1) = \rho(ev_2)$*
- *$\rho(ev_1) = \rho(ev_2)$ implies $o_1 = o_2$ and $\tilde{\mathsf{a}}_{1|\lambda} = \tilde{\mathsf{a}}_{2|\lambda}$*

Example 5. Coming back to our running example, consider a trace $T \in proj_0(\mathcal{C}^2)$ that executes a single passive session between the passport and the reader in S_{left}, the trace T would satisfy three events: $ev_1 = repl(o_3[1, 1])$ (unfolding of P_{left}), $ev_2 = repl(o_4[1, 1])$ (unfolding of R_{left}), and $ev_3 = repl_i(o_4[1, 1], n_l[1, 1])$ (unfolding of R_{left} with $n_l[1, 1]$ the message input by R_{left} and sent by P_{left}).

A session matching from P_{left} to P_{right} could be the function ρ such that $\rho(ev_1) = o_1[1]$ and $\rho(ev_2) = \rho(ev_3) = o_2[1]$ which would also correspond to a single session between the passport and the reader in S_{right}.

6.3 Falsifying conditions

Following Definition 10, we need to prove that for all traces T of $proj_0(\mathcal{C})$, there exists a convergent bitrace T' of $\mathcal{C}$ such that $proj_0(T') = T$. To prove that such a bitrace T' exists, we will show that we can build a bitrace such that none of the clauses in $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}))$ with bad as conclusion would have hypotheses matching T'. Thanks to Corollary 1, this will allow us to conclude that T' is convergent. From a clause $C = (H \to \mathrm{bad})$ of $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}))$, we generate two *falsifying conditions*, denoted $\mathsf{falsify}_0(C)$ and $\mathsf{falsify}_1(C)$, intuitively indicating sufficient conditions for a bitrace T' with $proj_0(T')$ (resp. $proj_1(T')$) satisfying $proj_0(H)$ (resp. $proj_1(H)$) to ensure that $proj_1(T')$ (resp. $proj_0(T')$) does not satisfy $proj_1(H)$ (resp. $proj_0(H)$).

Given a formula ϕ and a set of variables X, we denote by $\phi_{|X}$ the formula where we replace all variables of ϕ that are not in X by fresh names; thus only keeping variables from X. Since formulas in clauses are only composed of disequalities, we have the following property:

$$\text{For all substitutions } \sigma, \qquad \sigma \models \phi \qquad \text{implies} \qquad \sigma \models \phi_{|X}$$

In particular, we will use $\phi_{|\mathcal{X}_\lambda}$, where $\mathcal{X}_\lambda$ is the set of session identifier variables, in order to obtain formulas that constraint only session identifier variables.

Additionally, given a biclause $C = H \wedge \phi \rightarrow \text{bad}$, we write $\mathit{vars}_i(C)$ the set of variables in the i-th projection of H, i.e. $\mathit{vars}(\mathit{proj}_i(H))$.

Definition 17. *Let $i \in \{0, 1\}$. Given a fact F and a formula ϕ, we define the i-*matching conditions of F with ϕ*, denoted* $\mathsf{falsify}_i(F, \phi)$*, as the pair* (M, ϕ') *such that if* $F = \text{ev}(ev_0, ev_1)$ *is a replication bi-event with* $\mathsf{orepl}(ev_{1-i}) = o_{1-i}[\tilde{\mathsf{a}}]$ *then*

$$\mathsf{M} = \{(ev_i, \phi_{|fv(ev_i)}, y)\} \textit{ with } y \textit{ fresh and } \phi' = (y \neq o_{1-i}[\tilde{\mathsf{a}}_{|\lambda}])$$

otherwise $\mathsf{M} = \emptyset$ *and* $\phi' = \bot$.

We extend the notion of first and second matching conditions of a Horn clause as follows.

Definition 18. *Let C be the clause $C = (F_1 \ldots \wedge F_n \wedge \phi \rightarrow \text{bad})$. Let $i \in \{0, 1\}$. Let us denote by $\mathsf{falsify}_i(F_j, \phi) = (\mathsf{M}_j, \phi_j)$ for $j \in \{1, \ldots, n\}$. The i-matching conditions of C, denoted $\mathsf{falsify}_i(C)$, is defined as the tuple (H, M, ϕ') where*

- $\mathsf{X}_k = \bigcup_{j=1}^{n} \mathit{fv}(\mathit{proj}_k(F_j))$ *for* $k = 0, 1$
- $H = \mathit{proj}_i(F_1) \wedge \ldots \wedge \mathit{proj}_i(F_n) \wedge \phi_{|\mathsf{X}_i}$
- $\mathsf{M} = \bigcup_{j=1}^{n} \mathsf{M}_j$
- $\phi' = \forall \mathsf{X}_{1-i} \setminus \mathsf{X}_i.(\phi_1 \vee \ldots \vee \phi_n \vee \neg\phi_{|\mathcal{X}_\lambda \cup \mathsf{X}_{1-i}})$

Definition 19 (Falsifying condition)**.** *Let $C = (\phi \wedge H \wedge \bigwedge_{i=1}^{n} F_i \rightarrow \text{bad})$ be a clause such that the F_is are replication events and H contains any other facts. Let $k \in \{0, 1\}$. The k-*falsifying conditions *of C,* $\mathsf{falsify}_k(C)$*, is (Ω, ϕ') where:*

- *Ω is the function $[\mathit{proj}_k(F_i) \mapsto y_i]_{i=1}^{n}$ where the y_is are fresh distinct variables*
- $\phi' = \forall \tilde{x}_{1-k} \setminus \tilde{x}_k.(\neg\phi_{|\tilde{x}_{1-k|\lambda}} \vee \bigvee_{i=1}^{n} y_i \neq o_i[\tilde{\mathsf{a}}_{i|\lambda}])$

where $\mathsf{orepl}(\mathit{proj}_{1-k}(F_i)) = o_i[\tilde{\mathsf{a}}_i]$ *for $i = 1 \ldots n$ and $\tilde{x}_j = \mathit{vars}_j(C)$ for $j = 0, 1$*

Intuitively, when $\mathsf{falsify}_0(C) = (\Omega, \phi)$, the function Ω can be seen as a session matching on *open* terms that contain variables. The formula ϕ represents sufficient conditions for the hypotheses of the biclause C to be falsified. Therefore, any session matching that is an instantiation of Ω and that verifies ϕ will falsify the hypotheses of C.

Example 6. Consider the clauses C_1 and C_2 from Example 4. We have $\mathsf{falsify}_0(C_1) = (\Omega_1, \phi_1)$ where:

- $\Omega_1(\mathit{repl}(o_3[i_1, j_1])) = y_1$
- $\Omega_1(\mathit{repl}_i(o_4[i_2, j_2], n'[i_1, j_1])) = y_2$
- $\phi_1 = \forall i.(y_1 \neq o_1[i] \vee y_2 \neq o_2[i])$

and $\mathsf{falsify}_0(C_2) = (\Omega_2, \phi_2)$ where

- $\Omega_2(\mathit{repl}(o_3[i, j])) = y_1$
- $\Omega_2(\mathit{repl}_i(o_4[i, j'], n'[i, j])) = y_2$

- $\phi_2 = \forall i_1, i_2.(y_1 \neq o_1[i_1] \vee y_2 \neq o_2[i_2] \vee i_1 = i_2)$

Using Corollary 1, we can show that a session matching agreeing with a bitrace T and that falsifies all hypotheses of clauses deriving bad is necessary convergent:

Definition 20 (Falsification). *Let $\mathcal{C} = (\{\!\!\{(P_0, P_1)\}\!\!\}, \mathcal{A}, \emptyset)$ be an initial biconfiguration. Let $k \in \{0,1\}$. Let $T \in \mathsf{wrtrace}(proj_k(\mathcal{C}))$. Let $C = H \rightarrow \mathrm{bad}$ be a clause with $\mathsf{falsify}_k(C) = (\Omega, \phi)$.*

We say that T and a session matching ρ from P_k to P_{1-k} satisfies $\mathsf{falsify}_k(C)$, denoted $T, \rho \vdash_{cs} \mathsf{falsify}_k(C)$ when for all substitutions σ, if $T \vdash H\sigma$ and for all $ev \in dom(\Omega)$, $ev\sigma \in dom(\rho)$ and $\Omega(ev)\sigma = \rho(ev\sigma)$ then $\sigma \models \phi$.

Lemma 5 (Convergence criterion). *Let $\mathcal{C} = (\{\!\!\{(P_0, P_1)\}\!\!\}, \mathcal{A}, \emptyset)$ be an initial instrumented biconfiguration. Let $\mathbb{C}_{sat} = \mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}))$. Let $i \in \{0,1\}$. Let ρ be a matching mapping from P_i to P_{1-i}. Let $T \in \mathsf{wrtrace}^2(\mathcal{C})$.*

If the following properties hold:

1. *for all $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T \vdash^2 (ev_0, ev_1)$ and $\mathsf{orepl}(ev_{1-i}) = o[\tilde{\mathsf{a}}]$ implies $ev_i \in dom(\rho)$ and $\rho(ev_i) = o[\tilde{\mathsf{a}}_{|\lambda}]$*
2. *for all $C = (H \rightarrow \mathrm{bad}) \in \mathbb{C}_{sat}$, $proj_i(T), \rho \vdash_{cs} \mathsf{falsify}_i(C)$*

then T converges.

Proof. We prove the lemma by contradiction. Hence let us assume that T does not converge. By Corollary 1, we know there exists a clause $C = (H \wedge \phi \rightarrow \mathrm{bad})$ in $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}))$ and a substitution σ such that $\sigma \models \phi$ and for all $\mathrm{ev}(ev_0, ev_1) \in H$, $T \vdash^2 (ev_0\sigma, ev_1\sigma)$. By hypothesis, $proj_i(T), \rho \vdash_{cs} \mathsf{falsify}_i(C)$ holds. In particular, let us show how to build σ' that satisfies the hypothesis of Definition 20.

Let us denote $H = H' \wedge F_1 \wedge \ldots \wedge F_n$ with F_is the replication events. Let $\mathsf{falsify}_i(C) = (\Omega, \phi_m)$. We denote $H_m = proj_i(H \wedge \phi)$ and $\mathsf{X}_i = vars_i(C)$ Moreover, by definition, for all $ev \in dom(\Omega)$, if $\Omega(ev) = y$ and $\phi_{ev} = \phi_{|vars(ev)}$ then there exists $\mathrm{ev}(ev_0, ev_1) \in H$ such that $ev = ev_i$. Therefore, $T \vdash^2 (ev_0\sigma, ev_1\sigma)$ and by construction of H_m, we have $proj_i(T) \vdash H_m\sigma$. By item 1 of our hypothesis, we deduce that if $ev_i \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ and $\mathsf{orepl}(ev_{1-i}) = o[\tilde{\mathsf{a}}]$ then $ev_i \in dom(\rho)$ and $\rho(ev_i) = o[\tilde{\mathsf{a}}_{|\lambda}]$.

We define $\sigma' = \sigma \cup \{y \mapsto \rho(ev\sigma) \mid ev \in dom(\Omega) \wedge y = \Omega(ev)\}$. First, notice that $\sigma \models \phi$ implies $\sigma \models \phi_{\mathsf{X}_i}$ and so $\sigma' \models \phi_{\mathsf{X}_i}$. Moreover, as we already proved that $proj_i(T) \vdash H_m\sigma$, we deduce that $proj_i(T) \vdash H_m\sigma'$. Let $ev \in dom(\Omega)$ with $\Omega(ev) = y$ and $\phi_{ev} = \phi_{|vars(ev)}$, if $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ then by definition of σ', we have that $ev\sigma' \in dom(\rho)$ and $y\sigma' = \rho(ev\sigma')$.

Since $proj_i(T), \rho \vdash_{cs} \mathsf{falsify}_i(C)$ holds, we deduce that $\sigma' \models \phi_m$. Let us look at how ϕ_m is built:

$$\phi_m = \forall \mathsf{X}_{1-i} \setminus \mathsf{X}_i.(\phi_1 \vee \ldots \vee \phi_n \vee \neg\phi_{|\mathsf{X}_{1-i|\lambda}})$$

where for all $j \in \{1, \ldots, n\}$, $\phi_j = y_i \neq o_i[\tilde{\mathsf{a}}_{i|\lambda}]$ with $\mathsf{orepl}(proj_{1-i}(F_k)) = o_k[\tilde{\mathsf{a}}_k]$ for $k = 1 \ldots n$ and $\tilde{x}_j = vars_j(C)$ for $j = 0, 1$.

Notice that the free variables of ϕ_m are included in $\mathsf{X}_i \cup \{y_1, \ldots, y_n\}$. However, $dom(\sigma')$ contains all variables in $\phi_1 \vee \ldots \vee \phi_n \vee \neg\phi_{|\mathsf{X}_{1-i|\lambda}}$. Thus, $\sigma' \models \phi_m$ implies $\sigma' \models \phi_1 \vee \ldots \vee \phi_n \vee \neg\phi_{|\mathsf{X}_{1-i|\lambda}}$. But $\sigma \models \phi$ and so by definition of σ', we have $\sigma' \models \phi$ and so $\sigma' \models \phi_{|\mathsf{X}_{1-i|\lambda}}$. Hence, $\sigma' \models \phi_1 \vee \ldots \vee \phi_n$. However, we also know that when $\phi_j \neq \bot$, $\phi_j = (y_j \neq o[\tilde{\mathsf{a}_j}_{|\lambda}])$. But by item 1 of our hypotheses and by construction of σ', $y_j\sigma' = o[\tilde{\mathsf{a}_j}_{|\lambda}]$ which would imply that $\sigma' \models \bot$, hence a contradiction. We therefore conclude that T converges. □

Main results Lemma 5 is the core lemma that allows us to prove the different predicates. In the next three theorems, we consider $\mathcal{C}_I = (\{\!\!\{(P,Q)\}\!\!\}, \mathcal{A}, \emptyset)$ an initial biconfiguration and $\mathbb{C}_{sat}^{\text{bad}}$ the restriction of $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I))$ to clauses deriving bad.

Theorem 3 (May-testing preorder)**.** *If for all $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, there exists a session matching ρ from P to Q such that:*

- $\{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev\} \subseteq dom(\rho)$
- *for all $C \in \mathbb{C}_{sat}^{\text{bad}}$, $T, \rho \vdash_{cs} \mathsf{falsify}_0(C)$*

then $\pi_{\sqsubseteq_m}(\mathcal{C}_I)$ holds.

Theorem 4 (Observational preorder)**.** *If there exists a session matching ρ from P to Q such that:*

- $dom(\rho) = \{ev \in \mathsf{Ev}_{!o} \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))\}$
- *for all $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, if $\rho' = \rho \cup [repl(\bar{o}) \mapsto \rho(ev) \mid T \vdash repl_i(\bar{o}, M) = ev]$ then for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $T, \rho' \vdash_{cs} \mathsf{falsify}_0(C)$*

then $\pi_{\sqsubseteq_o}(\mathcal{C}_I)$ holds.

Theorem 5 (Observational equivalence)**.** *If there exists two matching mapping ρ_0 and ρ_1 from P to Q and Q to P respectively such that for all $i \in \{0,1\}$,*

- $dom(\rho_i) = \{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))\}$
- *for all $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$, for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $T, \rho_i \vdash_{cs} \mathsf{falsify}_i(C)$*
- *for all $ev_0 \in dom(\rho_0)$, for all $ev_1 \in dom(\rho_1)$, if $\mathsf{orepl}(ev_0) = o_0[\tilde{\mathsf{a}}_0]$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ then $o_0[\tilde{\mathsf{a}}_{0|\lambda}] = \rho_1(ev_1)$ iff $o_1[\tilde{\mathsf{a}}_{1|\lambda}] = \rho_0(ev_0)$*

then $\pi_{\approx_o}(\mathcal{C}_I)$ holds.

7 Practical Verification of Equivalences

7.1 Skolemisation

To prove the equivalences by relying on Theorems 3 to 5, we need to generate session matchings falsifying the hypotheses of all biclauses deriving bad. Although these session matchings may concretely depend on the considered traces, such dependencies are tedious to capture in practice. We therefore implement a stronger but simpler condition where, given a biclause C and its falsifying condition $\mathsf{falsify}(C) = (\Omega, \phi)$, we build *one* session matching satisfying $\mathsf{falsify}(C)$ for *any* instantiation of $dom(\Omega)$. In fact, if we denote $dom(\Omega) = \{$
$ev_i\}_{i=1}^n$ with $y_i = \Omega($
$ev_i)$, the satisfiability of $\mathsf{falsify}(C)$ can intuitively be interpreted as a first-order formula:

$$\forall ev_1 \ldots, \forall ev_n . \exists y_1 \ldots, \exists y_n . \phi$$

Building on this intuition, we will get rid of existential quantifiers via a *Skolemisation* process formalised below. To that end, we consider an additional set of name $\mathcal{N}_s$ that will be used as Skolem functions.

Definition 21 (Skolemisation)**.** *Let $\mathcal{C}^2$ be an initial biconfiguration. Let $C = (H \wedge \phi \rightarrow \text{bad})$ be a biclause. Let $i \in \{0,1\}$. Assume that $\mathsf{falsify}_i(C) = (\Omega, \phi')$. We say that a substitution σ is a* Skolemisation *of $\mathsf{falsify}_i(C)$ when $dom(\sigma) = img(\Omega)$, $\phi_{|vars_i(C)} \models \phi'\sigma$ and for all $ev, ev' \in dom(\Omega)$,*

- *if $\mathsf{orepl}(ev) = o_i[\tilde{\mathsf{a}}]$ then*

$$\Omega(ev)\sigma = o_{1-i}[s_1[\tilde{\mathsf{a}}_1], \ldots, s_k[\tilde{\mathsf{a}}_k]]$$

  with $(o_0, o_1) \in \mathrm{pm}(\mathcal{C}^2)$, $k = \mathsf{ar}^{\lambda}_{\mathcal{C}^2}(o_{1-i})$, $s_1, \ldots, s_k \in \mathcal{N}_s$ and $\tilde{\mathsf{a}}_1, \ldots, \tilde{\mathsf{a}}_k$ only contain subterms of ev

- $\mathsf{orepl}(ev) \preceq_{\mathcal{C}^2} \mathsf{orepl}(ev')$ *iff* $\Omega(ev)\sigma \preceq_{\mathcal{C}^2} \Omega(ev')\sigma$

Note that given a biclause concluding bad, there is only a finite number of possible Skolemisations of $\mathsf{falsify}_i(C)$ (modulo renaming of the names from $\mathcal{N}_s$). Similarly, in the following theorems, we will need to test the existence of skolemisation substitutions for each clause in $\mathbb{C}_{sat}$ deriving bad that satisfy some properties. As $\mathbb{C}_{sat}$ is also finite, the number of tuple of skolemisation substitutions is also finite (modulo renaming of the names from $\mathcal{N}_s$). Our implementation typically compute all these possible skolemisation substitutions and attempt to verify the conditions of Theorems 6 to 8.

Example 7. Coming back to Example 6, we can consider the following Skolemisation substitutions σ_1 and σ_2 of $\mathsf{falsify}(C_1)$ and $\mathsf{falsify}(C_2)$ respectively:

- $\sigma_1 = \{y_1 \mapsto o_1[s_1[i_1, j_1]]; y_2 \mapsto o_2[s_2[i_2, j_2, i_1, j_1]]\}$
- $\sigma_2 = \{y_1 \mapsto o_1[s_3[i, j]]; y_2 \mapsto o_2[s_3[i, j]]\}$

where $s_1, s_2, s_3, s_4 \in \mathcal{N}_s$. Notice that $\models \phi_1\sigma_1$ and $\models \phi_2\sigma_2$.

Our theorems also relies on a simplification function of Horn clauses used in the saturation procedure of ProVerif (see [BCC22, Section 3.2.5]) that, given a clause C, either $C\!\downarrow$ returns a set of simpler clauses or $\bot$. The details of this function are out of the scope of this paper and we only use its following property: for all clauses $C = (H \rightarrow \text{bad})$, if $C\!\downarrow = \bot$ then for all traces T of the initial configuration, for all substitutions σ, $T \not\models H\sigma$. That is, no trace can satisfy the hypotheses of the clause.

In the next three theorems, we consider an initial biconfiguration $\mathcal{C}_I = (\{\!\!\{(P, Q)\}\!\!\}, \mathcal{A}, \emptyset)$ and $\mathbb{C}^{\text{bad}}_{sat}$ the restriction of $\mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I))$ to clauses deriving bad. Moreover, we denote by $\mathsf{same}(ev_0, ev_1)$ the pair of events $(ev'_0, ev'_1) = (repl(\mathsf{orepl}(ev_0)), repl(\mathsf{orepl}(ev_1)))$ when $ev_i = repl(\overline{o})$ and $ev_{1-i} = repl_i(\overline{o}', M)$ for some $i = 0, 1$; and we have $(ev'_0, ev'_1) = (ev_0, ev_1)$ otherwise.

Theorem 6 (May-testing preorder)**.** *If for all $C \in \mathbb{C}^{\text{bad}}_{sat}$, we can associate a skolemisation substitution σ_C of $\mathsf{falsify}_0(C)$ such that for all $C_0, C_1 \in \mathbb{C}$ (C_0, σ_{C_0} and C_1, σ_{C_1} are renamed such that they have distinct variables) with $\mathsf{falsify}_0(C_i) = (\Omega_i, \phi_i)$ and $H'_i = proj_0(H_i)$, for all $ev_i \in dom(\Omega_i)$ for $i = 0, 1$, we have:*

1. *if $(ev'_0, ev'_1) = \mathsf{same}(ev_0, ev_1)$, $\alpha = mgu(ev'_0, ev'_1)$ then:*

$$(H'_0 \wedge H'_1 \wedge \Omega(ev_0)\sigma_{C_0} \neq \Omega(ev_1)\sigma_{C_1} \rightarrow \text{bad})\alpha\!\downarrow = \bot$$

2. if $\alpha = mgu(\Omega(ev_0)\sigma_{C_0}, \Omega(ev_1)\sigma_{C_1})$ then:

$$(H'_0 \wedge H'_1 \wedge \mathsf{orepl}(ev_0)_{|\lambda} \neq \mathsf{orepl}(ev_1)_{|\lambda} \rightarrow \mathrm{bad})\alpha{\downarrow} = \bot$$

then $\pi_{\sqsubseteq_m}(\mathcal{C}_I)$ holds.

Theorem 7 (Observational preorder)**.** *If for all $C \in \mathbb{C}^{\mathrm{bad}}_{sat}$, we can associate a skolemisation substitution σ_C of $\mathsf{falsify}_0(C)$ such that for all $C_i = (H_i \wedge \phi'_i \rightarrow \mathrm{bad}) \in \mathbb{C}^{\mathrm{bad}}_{sat}$ (taking C_i and σ_{C_i} renamed) with $\mathsf{falsify}_0(C_i) = (\Omega_i, \phi_i)$, for all $ev_i \in dom(\Omega_i)$ with $\Phi_i = \phi'_{i|vars(ev_i)}$ for $i = 0, 1$, we have:*

1. if $(ev'_0, ev'_1) = \mathsf{same}(ev_0, ev_1)$, $\alpha = mgu(ev'_0, ev'_1)$ then:

$$(\Phi_0 \wedge \Phi_1 \wedge \Omega(ev_0)\sigma_{C_0} \neq \Omega(ev_1)\sigma_{C_1} \rightarrow \mathrm{bad})\alpha{\downarrow} = \bot$$

2. if $\alpha = mgu(\Omega(ev_0)\sigma_{C_0}, \Omega(ev_1)\sigma_{C_1})$ then:

$$(\Phi_0 \wedge \Phi_1 \wedge \mathsf{orepl}(ev_0)_{|\lambda} \neq \mathsf{orepl}(ev_1)_{|\lambda} \rightarrow \mathrm{bad})\alpha{\downarrow} = \bot$$

then $\pi_{\sqsubseteq_o}(\mathcal{C})$ holds.

In both theorems, the first condition intuitively verifies that a concrete event cannot be associated to two different replication patterns (otherwise the session matching we build would not be a function). The second condition verifies that the session matching we build is injective (as required by the fourth bullet point in Definition 16).

Note that the two theorems only differ on the clauses given to the simplification function. For may-testing, we only need to build one session matching for each trace T; hence, the events in the hypotheses of the clauses C_0, C_1 can be assumed to be satisfied by T. On the contrary, for the observational pre-order, we need to build one session matching that covers all traces T. Hence, the hypotheses of the clauses C_0, C_1 can only be assumed to be satisfied by two possibly different traces T_0, T_1, respectively. As the events in the hypotheses of C_0, C_1 are not satisfied by the same trace, we cannot give them as arguments to the simplification function—which explains why it only receives formulae as arguments in Theorem 7.

The last theorem shows how we prove observational equivalence. Contrarily to may-testing and observational pre-order where we relied on skolemisation, we require a stronger property that is a bijection between the replication occurrences of the two initial processes.

Theorem 8 (Observational equivalence)**.** *If there exists a bijection β_0 from the occurrence replication names of P to the ones of Q (we denote β_1 its inverse) such that for all $i \in \{0, 1\}$, for all $o_i, o'_i \in dom(\rho_i)$,*

1. $\beta_i(o_i) = o_{1-i}$ implies $(o_0, o_1) \in \mathrm{pm}(\mathcal{C})$ and $\mathsf{ar}^{\lambda}_{\mathcal{C}_I}(o_0) = \mathsf{ar}^{\lambda}_{\mathcal{C}_I}(o_1)$

2. $o_i \preceq_{\mathcal{C}_I} o'_i$ if and only if $\beta_i(o_i) \preceq_{\mathcal{C}_I} \beta_i(o'_i)$

and for all $C = (H \wedge \phi \rightarrow \mathrm{bad}) \in \mathbb{C}^{\mathrm{bad}}_{sat}$ such that we have $\mathsf{falsify}_i(C) = (\Omega, \phi')$,

3. if $\sigma = \{\Omega(ev) \mapsto \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}] \mid ev \in dom(\Omega) \wedge \mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]\}$ then $\phi_{|vars_i(C)} \models \phi'\sigma$.

then $\pi_{\approx_o}(\mathcal{C})$ holds.

7.2 Experiments

Building on Theorem 6, we implemented a prototype for verifying may-testing. The source code is available in [Ano23b]. We fully reuse the specification language of ProVerif. Equivalence between two processes can be queried with the following command.

```
equivalence P Q
```

To activate our new feature, we added a new setting equivalenceRelation that can take five possible values: may−testing, simulation, observational, diff −equivalence (default value) and best. For example, may-testing can be queried by the following command.

```
set equivalenceRelation = may−testing .
```

When set to diff −equivalence, ProVerif will proceed as usual, that is, it will attempt to merge the two processes P and Q into a biprocess and prove diff-equivalence on it. When set to best, ProVerif will attempt to prove the strongest equivalence among observational, simulation and may-testing. On the running example, ProVerif yields the following results.

```
RESULT May−testing equivalence between process 1 and process 2 is true.
RESULT (but Process 2 simulated by process 1 is true.)
```

Table 1 displays the benchmarks we performed on several protocols from the literature. First, we included a couple of toy examples showcasing the may-testing, observational preorder and observational equivalence, as well as the lighter structural restrictions required by our approach compared to the baseline approach of ProVerif. We also verified the protocols included in ProVerif's distribution as sanity checks to ensure that our new algorithm does not induce a drop in expressivity and performance for previously provable protocols.

Second, our experiments include protocols such as BAC [For04] (various properties not limited to the minimal model of our motivation example), Hash-Lock [JW09], LAK [vDR08], PACE [BFK09, BDFK12], Helios [CGLM17], Feldhofer [FDW04]. Most of these models were taken from the Ukano tool that we compare against [HBD19].

Finally, we also show that our approach surpasses Ukano on, e.g., unlinkability proofs in a simplified standard TLS handshake. Indeed, Ukano requires that:

1. the protocol must be a two-party protocol between an initiator I and a responder R;

2. the processes of the initiator and responder cannot use the full range of ProVerif syntax, e.g., else-branches can only be null or restricted outputs of constants;

3. equivalences are syntactically restricted to be of the form:

$$!\mathsf{new}\ k; (!P_i \mid !P_R) \approx !\mathsf{new}\ k; (P_i \mid P_R)$$

4. the protocol must admit an appropriate *idealisation function* and must satisfy two sufficient conditions called *well-authentication* and *frame opacity*.

These conditions prevent Ukano to handle TLS. Indeed, a TLS handshake starts with a negotiation phase where the server and the client must agree on the TLS protocol version, the Diffie-Hellman (DH) groups and other cryptographic algorithms. During this phase, the server may send an HelloRetryRequest depending on the DH groups and key shares the client sent. Such requests subsequently affect the control flow of the protocol, which cannot be modelled with the restriction on else-branches discussed in Item 2. Besides, even if one considered a simpler version of TLS where Server and Client would have already agreed on the handshake parameters, the conditions 1 and 3 combined limit the security guarantees. In TLS, the identity of the client is typically its long term public key. In the above equivalence statements, k would thus plays the role of the private key associated to the public key of the client. Thus, Ukano could only prove unlinkability of TLS *without* revealing the public keys of the clients (as outputting them in P_I or P_R would trivially break unlinkability). Our approach can typically handle such models using a third outputting process in parallel. In fact, our experiment showed that on our simplified TLS models, even when considering no negotiation phase and no public key revealed, Ukano still fails to prove the two conditions discussed in Item 4.

Limitations To go beyond our simplified TLS models, we also tried to apply our prototype on the ProVerif models of [BCW22] which also proves unlinkability and anonymity of TLS clients. Their model describes in extensive details the protocol and consider some TLS extensions such as ECH and Pre-Shared Keys. Their proof is based on diff-equivalence but relies heavily on complex *restrictions* and manual reasoning that is, in our opinion, out of reach of standard ProVerif users. As such, with our prototype, we aimed to provide a fully automatic proof of client unlinkability that do not rely on such complex reasoning. However, our procedure showed a theoretical limitation as the TLS-ECH model in [BCW22] relies on global states such as tables to encode the Pre-Shared Keys. Although we do not prevent the use of tables, as soon as the protocol need to *desynchronise* their access, our procedure fails to show equivalence. Such problems also occur when the protocol relies on desynchronised private channels (e.g., BAC with a private channel to distribute the key between reader and passport). In practice, the TLS-ECH model in [BCW22] also raised issues by its size (several thousand lines of code) and our prototype could not complete the verification even after 48H of computation. There was already a bottleneck in the instrumentation and the clause generation.

8 Conclusion

We introduced new techniques to automatically verify process equivalences. As state-of-the-art tools for unbounded number of sessions are limited to the verification of *diff-equivalence*, our work is the first that is able to prove coarser-grained equivalences such as similarity and may-testing equivalence on syntactically unrestricted processes. We provided semantically sound conditions for proving these equivalences based on the set of saturated Horn clauses generated by ProVerif. We show how we satisfy these conditions in practice and implemented an extension of the tool.

Our work opens several directions for future work but, beyond those already discussed in the experimental section, one seems to be particularly interesting. One key hurdle (notably still pending even in the released version of ProVerif) is the handling of *trace restrictions* in equivalence proofs. In the context of may-testing, this would mean proving:

> *For all traces T_0 of P verifying φ_0, there exists an equivalent trace T_1 of Q verifying φ_1.*

where φ_0, φ_1 are user specified trace properties. Such double restrictions are naturally incompatible with refinement-based proofs as in ProVerif, i.e., that rely on a fine-grained relation

Protocol	Properties	Our Proofs	Time	U	PV
ProVerif distribution					
EKE	WeakSec.	✓$\approx_o$	2s	-	✓$\approx_o$
BAC (prv. ch)	Unlink.	✗		✗	✗
NSPK	StrongSec.	✓$\approx_o$	1s	-	✓$\approx_o$
Prv. Auth.	Anon.	✓$\approx_o$	1s	✗	✓$\approx_o$
WMF	StrongSec.	✓$\approx_o$	1s	-	✓$\approx_o$
Ukano distribution					
BAC+AA+PA	Anon.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
BAC+AA+PA	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
Feldhofer	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
Hash Lock	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
LAK	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
PACE	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	3m20s	✓$\approx_m$	✗
Simplified TLS					
Basic	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	2s	✗	✗
With HRR	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1m48s	-	✗
With HRR, PSK	Unlink.	✗		-	✗
Other models					
Running example	Unlink.	✓$\sqsubseteq_m \cap \sqsupseteq_o$	1s	✓$\approx_m$	✗
Helios	Vote Prv.	✓$\approx_o$	1s	-	✓$\approx_o$
Toy Simu 1,2		✓$\sqsubseteq_o \cap \sqsupseteq_o$	1s	-	✗
Toy Flow 1		✓$\sqsubseteq_o \cap \sqsupseteq_o$	1s	-	✗

Table 1: Benchmarks

✓$\mathcal{R}$: proof of the relation $\mathcal{R}$ -: syntactically not in the scope of the tool ✗: in the scope but the tool fails to prove

U: Results using Ukano PV: Results using ProVerif Vanilla

such as diff-equivalence to prove coarser ones. One possible direction would be to investigate whether an approach like ours directly targetting a coarser-grained, trace-based equivalence such as may-testing may circumvent such issues.

Index: Correspondence with CSF'23 Conference Paper

References

[ABF17] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 2017.

[Ano23a] Anonymised. Indistinguishability beyond diff-equivalence in proverif, 2023. Submitted to CSF'23.

[Ano23b] Anonymised. Indistinguishability beyond diff-equivalence in proverif: Source code and benchmarks. `https://www.dropbox.com/sh/t0bwuppbjab0dka/AADFc5E-zDlLPMyvF80fa1Sua?dl=0`, 2023. Submitted to CSF'23 as supplementary materials.

[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)*, 2017.

[BCC22] Bruno Blanchet, Vincent Cheval, and Véronique Cortier. Proverif with lemmas, induction, fast subsumption, and much more. In *Proceedings of the 43th IEEE Symposium on Security and Privacy (S&P'22)*. IEEE Computer Society Press, May 2022.

[BCD+19] David Basin, Cas Cremers, Jannik Dreier, Simon Meier, Ralf Sasse, and Benedikt Schmidt. *Tamarin prover manual.* available at `https://tamarin-prover.github.io/`, 2019.

[BCW22] Karthikeyan Bhargavan, Vincent Cheval, and Christopher A. Wood. A symbolic analysis of privacy for TLS 1.3 with encrypted client hello. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 365–379. ACM, 2022.

[BDFK12] Jens Bender, Özgür Dagdelen, Marc Fischlin, and Dennis Kügler. The PACE|aa protocol for machine readable travel documents, and its security. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, 2012.

[BDH+18] David A. Basin, Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5G authentication. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[BDM20] David Baelde, Stéphanie Delaune, and Solène Moreau. A method for proving unlinkability of stateful protocols. In *IEEE Computer Security Foundations Symposium (CSF)*, 2020.

[BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. Security analysis of the PACE key-agreement protocol. In *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, 2009.

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

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *CSF 2016*, 2016.

[BSCS20] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. *Automatic Cryptographic Protocol Verifier, User Manual and Tutorial.* available at `https://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf`, 2020.

[CB13] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with proverif. In *Proceedings of the International Conference on Principles of Security and Trust (POST)*, 2013.

[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)*, 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.

[CDD17] Véronique Cortier, Stéphanie Delaune, and Antoine Dallon. Sat-equiv: an efficient tool for equivalence properties. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17)*, 2017.

[CGCG$^+$18] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[CGLM17] Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. A type system for privacy properties. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017.

[CHH$^+$17] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[CKR18] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. DEEPSEC: Deciding equivalence properties in security protocols - theory and practice. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[CKR19] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. Exploiting symmetries when proving equivalence properties for security protocols. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[FDW04] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, 2004.

[For04] PKI Task Force. PKI for machine readable travel documents offering ICC read-only access. Technical report, International Civil Aviation Organization, 2004.

[HBD19] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for unbounded verification of privacy-type properties. *J. Comput. Secur.*, 2019.

[JW09] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. *ACM Trans. Inf. Syst. Secur.*, 13(1):7:1–7:23, 2009.

[KBB17] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017.

[SEMM14] Sonia Santiago, Santiago Escobar, Catherine Meadows, and José Meseguer. A formal definition of protocol indistinguishability and its verification using maude-npa. In *International Workshop on Security and Trust Management*, 2014.

[vDR08] Ton van Deursen and Sasa Radomirovic. Attacks on RFID protocols. *IACR Cryptol. ePrint Arch.*, page 310, 2008.

A Proof of Theorem 1

A.1 Link Between Regular and Instrumented Semantics

One important feature of the instrumented semantics is to compress (bounded and unbounded) parallel composition as a single indexed replication notation. We first extend the grammar of (regular) processes with a dummy construction $!_{\emptyset}P$, with the semantics

$$\{\!\{!_{\emptyset}P\}\!\} \cup \mathcal{P}, \Phi \to \{\!\{P\}\!\} \cup \mathcal{P}, \Phi \qquad \text{(Empty-Repl)}$$

The notions of may testing equivalence, inclusion, bisimilarity and similarity are also cast to this extended semantics. This introduces no changes in their decision in the sense of the following proposition:

Proposition 2. *If P is a process in the extended syntax, we write $P\downarrow$ the process in the regular syntax obtained by erasing all $!_{\emptyset}$ operators, i.e., by syntactically replacing all subprocesses of the form $!_{\emptyset}Q$ by Q. This notation is lifted to configurations by writing*

$$(\mathcal{P}, \Phi)\downarrow = (\{\!\{P\downarrow \mid P \in \mathcal{P}\}\!\}, \Phi)$$

Then $P \approx_o P\downarrow$ (and therefore $P \approx_m P\downarrow$).

Proof. It suffices to remark that the binary relation $\mathcal{S}$ on configurations defined by $A\mathcal{S}B$ *iff* $A\downarrow = B\downarrow$ is a bisimulation such that $P\mathcal{S}P\downarrow$. □

We also define a transformation $\llbracket\cdot\rrbracket$ of regular processes into this extended syntax, that introduces $!_\emptyset$ operators similarly as in instrumented processes. Formally:

$$\begin{aligned}
&\llbracket 0\rrbracket = 0\\
&\llbracket \alpha; P\rrbracket = \alpha; \llbracket P\rrbracket \qquad \text{for any instruction } \alpha\\
&\llbracket \mathsf{let}\ x = D\ \mathsf{in}\ P\ \mathsf{else}\ Q\rrbracket = \mathsf{let}\ x = D\ \mathsf{in}\ \llbracket P\rrbracket\ \mathsf{else}\ \llbracket Q\rrbracket\\
&\llbracket !P\rrbracket = !\llbracket P\rrbracket\\
&\llbracket P \mid Q\rrbracket = !_\emptyset\llbracket P\rrbracket \mid !_\emptyset\llbracket Q\rrbracket
\end{aligned}$$

Note that, by Proposition 2, we have for all processes P, Q that $P \approx Q$ *iff* $\llbracket P\rrbracket \approx \llbracket Q\rrbracket$, for any relation $\approx \in \{\approx_m, \sqsubseteq_m, \approx_o, \sqsubseteq_o\}$. Moving one step closer to the instrumented semantics, we also introduce the following notion of unfolding that describes a sequence of transitions excavating a process from nested parallel operators and replications.

Definition 22 (Unfolding). *Let $A = \{\!\{P\}\!\} \cup \mathcal{P}, \Phi$ be a configuration. An* unfolding *of P (in A) is a trace $t = t_1 \cdot t_2$ of A such that:*

1. *t contains at least one application of either Rule* Repl *or* Empty-Repl*;*
2. *$t_1 : A \Rightarrow A_1$ with $A_1 = \mathcal{P}_1 \cup \mathcal{P}, \Phi$ and t_1 only consists of applications of Rules* Restr *and* Par *in P, and is maximal (i.e., these two rules cannot be applied in $\mathcal{P}_1, \Phi$);*
3. *if there exists $P' \in \mathcal{P}_1$ that does not start with a replication (either $!$ or $!_\emptyset$, then t_2 may be empty, in which case we say that t* resulted in *P';*
4. *if there exists a decomposition $\mathcal{P}_1 = \{\!\{!Q\}\!\} \cup \mathcal{P}'_1$ or $\mathcal{P}_1 = \{\!\{!_\emptyset Q\}\!\} \cup \mathcal{P}'_1$ (N.B. this is not exclusive with the previous case), then t_2 may be of the form*
$$t_2 : A_1 \to A'_1 \Rightarrow A_2$$
where $A'_1 = (\{\!\{!Q, Q\}\!\} \cup \mathcal{P}'_1, \Phi)$ or $A'_1 = (\{\!\{Q\}\!\} \cup \mathcal{P}'_1, \Phi)$, i.e., the first transition of t_2 is obtained by applying either Rule Repl *or* Empty-Repl*, and the trace $A'_1 \Rightarrow A_2$ is an unfolding of Q in A_1.*

Note that the unfolded process P' is unique if $P = \llbracket P_0\rrbracket$ for some process P_0. We then define two additional semantics of regular processes, taking the form of two labelled transition relations $\xrightarrow{\alpha}_u$ and $\xrightarrow{\alpha}_{wu}$:

Definition 23 (Unfolding semantics). *We define $\xrightarrow{\alpha}_u$ as the transition relation on regular processes (extended with the dummy replication $!_\emptyset$) defined by taking the usual rules (Figure 1 and Rule* Empty-Repl*), but removing Rules* Restr*,* Par*,* Repl*, and* Empty-Repl*, and adding instead the following rule:*

$$\{\!\{P\}\!\} \cup \mathcal{P}, \Phi \to_u A \qquad \textit{if } \{\!\{P\}\!\} \cup \mathcal{P}, \Phi \Rightarrow A \textit{ is an unfolding of } P \textit{ in } A \qquad (\textsc{Unfold})$$

We define the extended notation $\overset{w}{\Rightarrow}_u$ as usual. We define $\xrightarrow{\alpha}_{wu}$ as the relation $\to_u$, except that the Unfold *is replaced by:*

$$A \xrightarrow{\alpha}_{wu} A' \qquad (\textsc{Weak-Unfold-In})$$

if $A \to_u A''$ by Rule Unfold, resulting in a process $P = \mathsf{in}(u, v); P'$, and $A'' \xrightarrow{\alpha} A'$ by either Rules In or Comm involving the toplevel input of P

$$A \to_{wu} A' \qquad (\textsc{Weak-Unfold})$$

if $A \to_u A'$ by Rule Unfold but Weak-Unfold-In is not applicable

In the next sections, we call *(weak) unfolding may testing equivalence, observational equivalence* the analogue versions of the usual process relations defined using the transition relations $\xrightarrow{\alpha}_u$ or $\xrightarrow{\alpha}_{wu}$. The relation between the instrumented semantics and the regular semantics can then be formalised as follows, using the unfolding semantics.

Lemma 6. *Let P be a process, $A_0 = [\![P]\!]$, and $t_u : A_0 \xrightarrow{\alpha_1}_u \cdots \xrightarrow{\alpha_n}_u A_n$ be an unfolding trace. We also let $B_0 = [\![P]\!]_i$. Then there exists an instrumented trace of the form*

$$t_I : B_0 \Rightarrow_i B_0' \xrightarrow{\beta_1}_i B_1 \Rightarrow_i B_1' \xrightarrow{\beta_2}_i \cdots \Rightarrow_i B_n \xrightarrow{\beta_n}_i B_n'$$

such that there exists a bijection ρ from name patterns to names such that the following properties hold.

1. *All traces $B_i \Rightarrow_i B_i'$ only consist of transitions* I-Par, I-App *and* I-Gen, *and the traces $B_i' \xrightarrow{\beta_{i+1}}_i B_{i+1}$ are never derived from either of these three rules. In addition, Rule* I-Par *should not be applicable from B_i'.*

2. *The transition $B_i' \xrightarrow{\beta_{i+1}}_u B_{i+1}$ is derived by Rule* I-Nil *(resp.* I-Repl, I-Comm, I-Let1, I-Let2, I-Out, I-In, I-Event*) iff the transition $A_i \xrightarrow{\alpha_{i+1}}_i A_{i+1}$ is derived by Rule* Nil *(resp.* Unfold, Comm, Let1, Let2, Out, In, Event*). In the cases of the rules* I-In, I-Out *and* I-Comm, *the input/output term M of the instrumented trace is so that the input/output term of the unfolding trace is $M\sigma$.*

3. *For all frames Φ such that $A_i = \mathcal{P}, \Phi$, and for all $w \in dom(\Phi)$, $w\Phi \in \mathcal{A}$ with $B_i = \mathcal{P}', \mathcal{A}, \Lambda$.*

4. *For all sets $\mathcal{A}$ such that $B_i' = \mathcal{P}', \mathcal{A}, \Lambda$, for all $M \in \mathcal{A}$, there exists a recipe ξ such that $\xi\Phi \Downarrow M$ with $A_i = \mathcal{P}, \Phi$.*

An analogue statement (omitted there for succinctness) can be derived for the weak variants, i.e., replacing $\xrightarrow{\alpha}_u$ by $\xrightarrow{\alpha}_{wu}$ and $\xrightarrow{\alpha}_i$ by $\xrightarrow{\alpha}_{wi}$. Conversely, for all instrumented traces of the form of t_I, there exists an unfolding trace of the form t_u verifying the points above.

A.2 Convergence and May Testing

We prove in this section Theorem 1 in the case of may testing inclusion $\sqsubseteq_m$. The same result for may testing equivalence $\approx_m$ is then a direct corollary, observing that $\pi_{\sqsubseteq_m} \cap \pi_{\sqsupseteq_m} = \pi_{\approx_m}$. To prove the desired result, we introduce a notion of permutability of trace actions that will allow to reorganise traces to meet the specific requirements of instrumented traces (i.e., to obtain unfolding traces).

Definition 24 (independence). *Let $A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n$ be a trace, and $i \in [\![0, n-2]\!]$. We say that the two transitions $t_i : A_i \xrightarrow{\alpha_{i+1}} A_{i+1}$ and $t_{i+1} : A_{i+1} \xrightarrow{\alpha_{i+2}} A_{i+2}$ are* independent *if they can be written under the form*

$$t_i : (\mathcal{P}_i \cup \mathcal{Q}_i, \Phi) \xrightarrow{\alpha_{i+1}} (\mathcal{P}_i' \cup \mathcal{Q}_i, \Phi \cup \Phi')$$
$$t_{i+1} : (\mathcal{P}_i' \cup \mathcal{Q}_i, \Phi \cup \Phi') \xrightarrow{\alpha_{i+2}} (\mathcal{P}_i' \cup \mathcal{Q}_i', \Phi \cup \Phi' \cup \Phi'')$$

where, in case $\alpha_{i+2} = \mathsf{in}(\xi, \zeta)$ is an input action and $\alpha_{i+1} = \mathsf{out}(\xi, \omega)$ an output action, the axiom ω should appear neither in ξ nor ζ.

This notion of independence is standard when developing, for instance, partial-order reductions for trace or indistinguishability properties [**?**, CKR19]. Its main property of interest is the permutability of independent transitions.

Proposition 3 (Independent permutability). *Using the notations of the above definitions, if* t_i *and* t_{i+1} *are independent, then there exists a trace of the form:*

$$A_0 \xRightarrow{\alpha_1 \cdots \alpha_i} A_i \xRightarrow{\alpha_{i+2}\alpha_{i+1}} A_{i+2} \xRightarrow{\alpha_{i+3} \cdots \alpha_n} A_n$$

Writing τ *the permutation of* $[\![1, n]\!]$ *such that* $\tau(i+1) = i+2$, $\tau(i+2) = i+1$ *and* $\tau(k) = k$ *if* $k \notin \{i+1, i+2\}$*, this trace is referred to as* $\tau.t$*. This notation is extended to instrumented traces and bitraces in the natural way.*

The proof of this proposition is immediate. More generally, we may use the notation $\tau.t$ for successive applications of this group action, i.e., $(\tau_1 \circ \tau_2).t$ refers to $\tau_1.(\tau_2.t)$. In this case we call τ a *permutation of independent transitions* of t. A simple corollary of this proposition is the following one:

Corollary 2 (Unfoldability). *Let* P *be a process and* t *be a trace of* $[\![P]\!]$ *(in the sense of Appendix A.1). Then there exists an extension* $t_e = t \cdot t'$ *of* t *and a permutation* π *of independent transitions of* $t \cdot t'$ *such that:*

1. t' *only consists of applications of Rules* Restr*,* Par*,* Repl*,* Comm *or* In*;*

2. $\pi.t_e$ *can be interpreted (in the natural way) as a* $\rightarrow_{wu}$*-trace.*

Intuitively, this results expresses that traces can be interpreted as unfolding traces up to minor modifications not affecting proofs of may testing equivalence. This is formally captured by the proof arguments for Theorem 1.

▷ *Proof of Theorem 1, case* $\sqsubseteq_m$. Let P, Q be two processes, and $\mathcal{C} = (\{\!\{([\![P]\!]_i, [\![Q]\!]_i)\}\!\}, \emptyset, \emptyset)$. We assume that $\pi_{\sqsubseteq_m}(\mathcal{C})$ and we want to prove that $P \sqsubseteq_m Q$. By Proposition 2, it suffices to prove that $[\![P]\!] \sqsubseteq_m [\![Q]\!]$. Let thus t be a trace of $[\![P]\!]$ and S be a set of recipes (built from axioms appearing in t). By Corollary 2, we can let $t_e = t \cdot t'_e$ an extension of t such that t'_e only consists of Rules Restr, Par, Repl, Comm or In, as well as π permuting independent transitions of t_e such that $\pi.t_e$ can be interpreted as a $\rightarrow_{wu}$-trace t_u. For that we let t_I be the instrumented trace obtained by applying Lemma 6 to t_u, that we extend to $t_u \cdot t_u^A$, where t_u^A contains applications of Rules I-App and I-Gen necessary to construct all recipes of S. We then let t_I^2 be the convergent bitrace obtained by applying the assumption $\pi_{\sqsubseteq_m}(\mathcal{C})$. As the transitions of t'_e are either silent or do not affect the frame, and by convergence of $\mathcal{C}$, it suffices to take t' an adequate prefix of $\pi'.t'_I$, with t'_I the trace obtained by the converse of Lemma 6 applied to the instrumented trace $proj_1(t_I^2)$, and π' permuting independent transitions of t'_I. □

A.3 Convergence and Bisimilarity

We only establish the proof of Theorem 1 in the case of $\pi_{\approx_o}$, and the proofs for $\pi_{\sqsubseteq_o}$ can be obtained using an identical reasoning. Similarly to the proof above for may testing equivalence, we first establish results allowing to reduce the proofs of observational equivalence to their unfolding version.

Proposition 4. *Let P, Q be two processes. If $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are unfolding observationally equivalent, then P and Q are observationally equivalent.*

Proof. Let us assume that $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are weak unfolding observationally equivalent, and let us prove, by Proposition 2, that $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are observationally equivalent. We thus let $\mathcal{S}$ be an unfolding bisimulation such that $\llbracket P \rrbracket \mathcal{S} \llbracket Q \rrbracket$, and we define the relation $\bar{S}$ by $A\bar{\mathcal{S}}B$ *iff* there exist $A \Rightarrow A'$ and $B \Rightarrow B'$ two traces only consisting of Rules Restr, Repl, Par or Empty-Repl such that $A'\mathcal{S}B'$. Note that $\llbracket P \rrbracket \bar{\mathcal{S}} \llbracket Q \rrbracket$. Let us thus prove that $\bar{\mathcal{S}}$ is a bisimulation.

First of all, since $\mathcal{S}$ is a symmetric relation included in static equivalence, $\bar{\mathcal{S}}$ is one as well. Let then A, B be two configurations such that $A\bar{\mathcal{S}}B$ (and therefore $A\mathcal{S}B$), a transition $A \xrightarrow{\alpha} A'$, and let us construct $B \overset{\beta}{\Rightarrow} B'$ such that $\alpha \equiv \beta$ and $A'\bar{\mathcal{S}}B'$. The result directly follows from the fact that $A\mathcal{S}B$ if the transition $A \xrightarrow{\alpha} A'$ is derived from any rule except Restr, Repl, Par or Empty-Repl. If the transition is derived by either of these rules, we have $A'\bar{\mathcal{S}}B$ by definition, hence the result. □

Using a similar construction, we can show that $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ being weak unfolding observationally equivalent implies them being unfolding observationally equivalent. Hence:

Corollary 3. *Let P, Q be two processes. If $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are weak unfolding observationally equivalent, then P and Q are observationally equivalent.*

Altogether, we obtain the following proof:

▷ *Proof of Theorem 1, case $\pi_{\approx_o}$.* Let P, Q be two processes, and $\mathcal{C} = (\{\!\{(\llbracket P \rrbracket_i, \llbracket Q \rrbracket_i)\}\!\}, \emptyset, \emptyset)$. We assume that $\pi_{\approx_o}(\mathcal{C})$ and we want to prove that $P \approx_o Q$. By Corollary 3, it suffices to prove that $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are weak unfolding observationally equivalent. We define a relation $\mathcal{S}$ on processes as the smallest symmetric relation such that, for all biconfigurations $\mathcal{C}'$ such that $t_I^2 : \mathcal{C} \overset{w_I}{\Longrightarrow}_{i^2} \mathcal{C}'$ and $\pi_{\approx_o}(\mathcal{C}')$, writing $t_I = proj_0(t_I^2)$ (resp. $t_I = proj_1(t_I^2)$), then for all weak unfolding traces $t_u : \llbracket P \rrbracket \overset{w}{\Rightarrow} A$ (resp. $t_u : \llbracket Q \rrbracket \overset{w}{\Rightarrow} B$) verifying the properties of the converse of Lemma 6 w.r.t. t_I, we have $A\mathcal{S}B$. Note in particular that $\llbracket P \rrbracket \mathcal{S} \llbracket Q \rrbracket$. It therefore suffices to prove that $\mathcal{S}$ is a weak unfolding bisimulation to conclude. Let A, B be two configurations such that $A\mathcal{S}B$.

We first prove that A and B are statically equivalent. By definition, we let $\mathcal{C}'$ be a biconfiguration such that $t_I^2 : \mathcal{C} \overset{w_I}{\Longrightarrow}_{i^2} \mathcal{C}'$, $\pi_{\approx_o}(\mathcal{C}')$ and for two traces $\llbracket P \rrbracket \overset{w}{\Rightarrow} A$ and $\llbracket Q \rrbracket \overset{w}{\Rightarrow} B$ (resp. $\llbracket P \rrbracket \overset{w}{\Rightarrow} B$ and $\llbracket Q \rrbracket \overset{w}{\Rightarrow} A$, as $\mathcal{S}$ is defined as a symmetric closure) verifying the properties of the converse of Lemma 6 w.r.t. $proj_0(t_I^2)$ and $proj_1(t_I^2)$. In particular, writing $A = (\mathcal{P}, \Phi_0)$, $B = (\mathcal{Q}, \Phi_1)$, $proj_i(\mathcal{C}') = (\mathcal{P}_i, \mathcal{A}_i, \Lambda_i)$ we have $dom(\Phi) = dom(\Psi)$ and, by Item 3, for all $w \in dom(\Phi_i)$, $w\Phi_i \in \mathcal{A}_i$. Let then ξ be a recipe and let us prove that $\xi\Phi_0 \Downarrow \mathsf{fail}$ *iff* $\xi\Phi_1 \Downarrow \mathsf{fail}$. We consider a sequence of I-App and I-Gen transitions from $proj_0(\mathcal{C}')$ (or $proj_1(\mathcal{C}')$ for the symmetric case, whose treatment is analogous), corresponding to the construction of the recipe $\xi\rho^{-1}$, where ρ is the bijection from name patterns to names given by Lemma 3 by hypotheses on A and B. Since $\pi_{\approx_o}(\mathcal{C}')$, we obtain the corresponding sequence of transitions $\mathcal{C}' \Rightarrow_{i^2} \mathcal{C}''$, where $\mathcal{C}''$ is convergent. In particular, we obtain the expected conclusion by using Item 9 of the definition of convergence (Definition 9).

Let us then assume in addition that $A \xrightarrow{\alpha}_{wu} A'$. We thus let an extension of the instrumented trace $proj_0(t_2^I)$ given by the converse of Lemma 6, written $proj_0(t_I^2) \cdot t_I$ where $t_I : proj_0(\mathcal{C}') \overset{\alpha}{\Rightarrow}_{wi} A''$. In particular, since $\pi_{\approx_o}(\mathcal{C}')$, there exists a convergent bitrace

$s_I^2 : \mathcal{C}' \overset{\alpha}{\Rightarrow}_{wi^2} \mathcal{C}''$ such that $proj_0(s_I^2) = t_I$. It then suffices to consider the weak unfolding trace $B \overset{\alpha}{\Rightarrow}_{wu} B'$ obtained by applying Lemma 6 to $proj_1(s_I^2)$, as then $A'\mathcal{S}B'$. □

B Properties on occurrences

We extend the notation $\prec_\mathcal{C}$ to all occurrence names (including the one on inputs).

Lemma 7. *Let $\mathcal{C}$ be an initial configuration. For all o, o' occurrence names in $\mathcal{C}$, if $o \prec_\mathcal{C} o'$ and*

- *either $!^{o[\tilde{\mathsf{a}}]}_{\tilde{\mathsf{b}}} P$ or $\mathsf{in}^{o[\tilde{\mathsf{a}}]}(M, x); P$ occurs in $\mathcal{C}$*
- *and either $!^{o'[\tilde{\mathsf{a}}']}_{\tilde{\mathsf{b}}'} P$ or $\mathsf{in}^{o'[\tilde{\mathsf{a}}']}(M, x); P$ occurs in $\mathcal{C}$*

then $o[\tilde{\mathsf{a}}] \prec_\mathcal{C} o'[\tilde{\mathsf{a}}']$. Moreover, if o and o' are two occurrence replication names then $\tilde{\mathsf{a}}$ is a prefix of $\tilde{\mathsf{a}}'$. Finally, if $\tilde{\mathsf{a}}$ is ground then $o[\tilde{\mathsf{a}}]$ can only occur once

Proof. Directly given by the transformation from a process to an instrumented process. □

Lemma 8. *Let $\mathcal{C}$ be an initial instrumented configuration. Let $T \in \mathsf{wrtrace}(\mathcal{C})$. Let $i \in \mathbb{N}$ such that $T[i] = \mathcal{P}, \mathcal{A}, \Lambda$.*

For all occurrence names o, o' in $T[i]$ and $\{ev \mid T, j \vdash ev \wedge j \leq i\}$, if $o \prec_\mathcal{C} o'$ and

- *either $!^{o[\tilde{\mathsf{a}}]}_{\tilde{\mathsf{b}}} P$ or $\mathsf{in}^{o[\tilde{\mathsf{a}}]}(M, x); P$ occurs in $T[i]$ or $T, j \vdash ev$ with $j \leq i$ and $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$;*
- *and either $!^{o'[\tilde{\mathsf{a}}']}_{\tilde{\mathsf{b}}'} P$ or $\mathsf{in}^{o'[\tilde{\mathsf{a}}']}(M, x); P$ occurs in $T[i]$ or $T, j \vdash ev$ with $j \leq i$ and $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$;*

then $o[\tilde{\mathsf{a}}] \prec_\mathcal{C} o'[\tilde{\mathsf{a}}']$. Moreover, if o and o' are two occurrence replication names then $\tilde{\mathsf{a}}$ is a prefix of $\tilde{\mathsf{a}}'$. Finally, if $\tilde{\mathsf{a}}$ is ground then $o[\tilde{\mathsf{a}}]$ only occurs once in $T[i]$ and $\{ev \mid T, j \vdash ev \wedge j \leq i\}$.

Proof. The proof is done by induction on i. It relies on the following observation:

- $o[\tilde{\mathsf{a}}] \prec_\mathcal{C} o'[\tilde{\mathsf{a}}]$ implies that $o[\tilde{\mathsf{a}}]\sigma \prec_\mathcal{C} o'[\tilde{\mathsf{a}}]\sigma$ for all substitution σ.
- $T, j \vdash ev$ with $j \leq i$ and $\mathsf{orepl}(ev)$ an occurence replication name imply $\mathsf{orepl}(ev) \in \Lambda$.
- The rule I-Repl always instantiate the bound session identifier in such a way that $o[\tilde{\mathsf{a}}]$ was not already in Λ
- When the rule I-In or I-Comm is applied, the occurrence pattern $o[\tilde{\mathsf{a}}]$ is ground hence only occur on the process on which the rule is applied. □

Lemma 2. *Let $\mathcal{C}$ be an initial instrumented configuration. For all $i, j \in \mathbb{N}$, for all $T \in \mathsf{wrtrace}(\mathcal{C})$,*

- *if $T, i \vdash pre(\overline{o_1}, M)$ and $T, j \vdash pre(\overline{o_2}, N)$; or $T, i \vdash repl_i(\overline{o_1}, M)$ and $T, j \vdash repl_i(\overline{o_2}, N)$ then*
  1. *$\overline{o_1} = \overline{o_2}$ if and only if $i = j$*
  2. *$i = j$ implies $M = N$*

- *if* $T, i \vdash ev$ *and* $T, j \vdash ev'$ *with* $ev, ev' \in \mathsf{Ev}_{!}$ *then*

  1. $\mathsf{orepl}(ev) = \mathsf{orepl}(ev')$ *implies* $i = j$
  2. $i = j$ *implies* $ev = ev'$

- *if* $T \vdash ev$ *and* $T \vdash ev'$ *with* $ev, ev' \in \mathsf{Ev}_{!}$, $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$ *and* $\mathsf{orepl}(ev) = o[\tilde{\mathsf{b}}]$ *then* $\tilde{\mathsf{a}}_{|\lambda} = \tilde{\mathsf{b}}_{|\lambda}$ *implies* $\tilde{\mathsf{a}} = \tilde{\mathsf{b}}$.

Proof. Let $T \in \mathsf{wrtrace}(\mathcal{C})$ and $i, j \in \mathbb{N}$. Assume that $T, i \vdash pre(o_1[\tilde{\mathsf{a}}_1], M)$ and $T, j \vdash pre(o_2[\tilde{\mathsf{a}}_2], N)$. If $i = j$ then by definition of the semantics, we directly have that $pre(o_1[\tilde{\mathsf{a}}_1], M) = pre(o_2[\tilde{\mathsf{a}}_2], N)$, hence $M = N$ and $o_1[\tilde{\mathsf{a}}_1] = o_2[\tilde{\mathsf{a}}_2]$. If $o_1[\tilde{\mathsf{a}}_1] = o_2[\tilde{\mathsf{a}}_2]$ then we know from the semantics that $o_1[\tilde{\mathsf{a}}_1]$ is ground. Let us assume w.l.o.g. that $i \geq j$. By Lemma 8, we know that $o_1[\tilde{\mathsf{a}}_1]$ occur once in $T[i]$ and $\{ev \mid T, j' \vdash ev \wedge j' \leq i\}$. Thus, $i = j$. Similar proof allow holds when $T, i \vdash repl_i(o_1[\tilde{\mathsf{a}}_1], M)$ and $T, j \vdash repl_i(o_2[\tilde{\mathsf{a}}_2], N)$; and when $T, i \vdash repl(o_1[\tilde{\mathsf{a}}_1])$ and $T, j \vdash repl(o_2[\tilde{\mathsf{a}}_2])$ then we have once again by the semantics that $i = j$ implies $repl(o_1[\tilde{\mathsf{a}}_1]) = repl(o_2[\tilde{\mathsf{a}}_2])$. □

Lemma 3 (Consistency). *Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in \mathsf{wrtrace}^2(\mathcal{C})$, if $T, i \vdash^2 (ev_0, ev_1)$ and $T, j \vdash^2 (ev'_0, ev'_1)$ with $ev_0, ev_1, ev'_0, ev'_1 \in \mathsf{Ev}_{!} \cup \mathsf{Ev}_{!i}$ then*

- $(\mathsf{orepl}(ev_0), \mathsf{orepl}(ev_1)) \in \mathrm{pm}(\mathcal{C})$
- $\mathsf{orepl}(ev_0) \prec_{\mathcal{C}} \mathsf{orepl}(ev'_0)$ *if and only if* $\mathsf{orepl}(ev_1) \prec_{\mathcal{C}} \mathsf{orepl}(ev'_1)$

Proof. A simple proof by induction on the size of the trace T allows us to show that for all $i \in \mathbb{N}$, if $T[i] = \mathcal{P}, \mathcal{A}, \Lambda$ then for all $(P, Q) \in \mathcal{P}$, $\mathsf{pm}(P, Q) \subseteq \mathrm{pm}(\mathcal{C})$. Thus when the replication rule is executed, the two occurrence replication names are by definition in $\mathrm{pm}(\mathcal{C})$. The second bullet point follows from Lemma 8. □

Lemma 4. *Let $\mathcal{C}_0$ be an initial convergent instrumented biconfiguration. Let $\mathcal{C}_0 \overset{\mathsf{tr}}{\Rightarrow}_{i^2} \mathcal{C}_1 = \mathcal{P}, \mathcal{A}, \Lambda$. Let $i \in \{0, 1\}$.*

Assume $proj_i(\mathcal{C}_1) \xrightarrow{repl(o_i[\tilde{\mathsf{a}}_i])}_i \mathcal{C}'_2$. For all o_{1-i} such that $k = \mathsf{ar}^{\lambda}_{\mathcal{C}_0}(o_{1-i})$, for all $\lambda_1, \ldots, \lambda_k \in \mathbb{N}$, if $(o_0, o_1) \in \mathrm{pm}(\mathcal{C}_0)$ and for all $(o'_0[\tilde{\mathsf{a}}'_0], o'_1[\tilde{\mathsf{a}}'_1]) \in \Lambda$,

- $o'_i[\tilde{\mathsf{a}}'_i] \prec_{\mathcal{C}_0} \overline{o}_i$ *implies* $o'_{1-i}[\tilde{\mathsf{a}}'_{1-i}] \prec_{\mathcal{C}_0} o_{1-i}[\lambda_1, \ldots, \lambda_k]$
- $o'_{1-i}[\tilde{\mathsf{a}}'_{1-i|\lambda}] \neq o_{1-i}[\lambda_1, \ldots, \lambda_k]$

then $\mathcal{C}_1 \xrightarrow{(repl(o_0[\tilde{\mathsf{a}}_0]), repl(o_1[\tilde{\mathsf{a}}_1]))}_{i^2} \mathcal{C}_2$ for some $\tilde{\mathsf{a}}_{1-i}$ with $proj_i(\mathcal{C}_2) = \mathcal{C}'_2$ and $\tilde{\mathsf{a}}_{1-i|\lambda} = \lambda_1, \ldots, \lambda_k$.

Proof. Let us assume w.l.o.g. that $i = 0$ (the proof of $i = 1$ is symmetrical). Let us denote $\mathcal{C}'_2 = \mathcal{P}'_2, \mathcal{A}_2, \Lambda_2$.

Since $proj_0(\mathcal{C}_1) \xrightarrow{repl(o_0[\tilde{\mathsf{a}}_0])}_i \mathcal{C}'_2$, we deduce that $\mathcal{P} = \mathcal{P}' \cup \{\!\{(\{\!\{!^{\overline{o}}_{\tilde{\mathsf{b}}}P\}\!\} \cup \mathcal{M}, \mathcal{M}')\}\!\}$, $\mathcal{P}'_2 = proj_0(\mathcal{P}') \cup \{\!\{P\sigma, \{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}}P\}\!\} \cup \mathcal{M}\}\!\}$, $\Lambda_2 = proj_0(\Lambda) \cup \{o_0[\tilde{\mathsf{a}}_0]\}$ with $dom(\sigma) = \tilde{\mathsf{b}}$, $o_0[\tilde{\mathsf{a}}_0] = \overline{o}\sigma$ and $\overline{o}\sigma \notin proj_0(\Lambda)$.

By construction of $\mathrm{pm}(\mathcal{C}_0)$ and by application of Lemma 3, if $(o'_0[\tilde{\mathsf{a}}'_0], o'_1[\tilde{\mathsf{a}}'_1]) \in \Lambda$ such that o_0 is directly in the scope of $o'_0[\tilde{\mathsf{a}}'_0]$ then the only occurrence names o_1 such that $(o_0, o_1) \in \mathrm{pm}(\mathcal{C}_0)$ and $o'_1[\tilde{\mathsf{a}}'_1] \prec_{\mathcal{C}_0} o_1[\lambda_1, \ldots, \lambda_k]$ for some $\tilde{\mathsf{a}}_1$ are the occurrences at the root of $\mathcal{M}'$, i.e. $\mathcal{M} = \{\!\{!^{\overline{o'}}_{\tilde{\mathsf{b}}'}P'\}\!\} \cup \mathcal{M}''$ and σ' such that $\overline{o'}\sigma' = o_1[\tilde{\mathsf{a}}_1]$ and $\lambda_1, \ldots, \lambda_k = \tilde{\mathsf{a}}_{1|\lambda}$.

Since we also have that $o_1'[\tilde{\mathsf{a}}'_{1|\lambda}] \neq o_1[\lambda_1, \ldots, \lambda_k]$, by using Lemma 2, we deduce that $o_1[\tilde{\mathsf{a}}_1] \notin \mathit{proj}_1(\Lambda)$. Thus allows to us conclude that the application of the bistep I^2-Repl is possible. □

C Proof of Theorem 2

Definition 25. *Let $\mathcal{C}_0$ be an initial instrumented configuration. Let $T \in \mathsf{wrtrace}^2(\mathcal{C}_0)$ be a data compliant trace. Let $\mathcal{D}$ be a derivation of F at step τ. We say that T* satisfies a derivation *$\mathcal{D}$, denoted $T \vdash \mathcal{D}$, when for all nodes η of $\mathcal{D}$, if F_0 is the label of the incoming edge of η then*

1. *if $F_0 = \mathrm{att}(f(M_1, \ldots, M_m), f(M'_1, \ldots, M'_m))$ with $f \in \mathcal{F}_{data}$ then*
   - *either R is the clause $\mathrm{att}(x_1, x'_1) \wedge \ldots \wedge \mathrm{att}(x_m, x'_M) \rightarrow \mathrm{att}(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m))$;*
   - *or η is not the root and the node η' connected to the incoming edge of η is labelled with the clause $\mathrm{att}(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m)) \rightarrow \mathrm{att}(x_j, x'_j)$ for some j*
2. *if $F_0 = \mathrm{ev}(ev, ev')$ then $T \vdash (ev, ev')$*

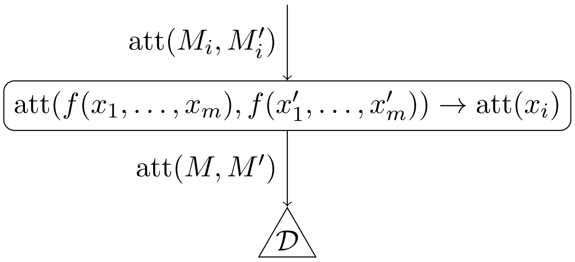
When all nodes but the root satisfy the two properties and the root satisfy only property 2, we say that T weakly satisfy $\mathcal{D}$, denoted $T \vdash_w \mathcal{D}$.

Lemma 9. *Let $\mathcal{C}_0 = P_0, \mathcal{A}_0$ be an initial instrumented configuration. Let $T \in \mathsf{wrtrace}^2(\mathcal{C}_0)$. Let $\mathbb{C} = \mathbb{C}_{\mathcal{A}}(\mathcal{C}_0) \cup \mathbb{C}_{\mathcal{P}}(\mathcal{C}_0) \cup \mathbb{C}_e(T)$. For all terms M, M', for all derivations $\mathcal{D}$ of $\mathrm{att}(M, M')$ such that $T \vdash_w \mathcal{D}$, there exists a derivation $\mathcal{D}'$ of $\mathrm{att}(M, M')$ such that $T \vdash \mathcal{D}'$.*

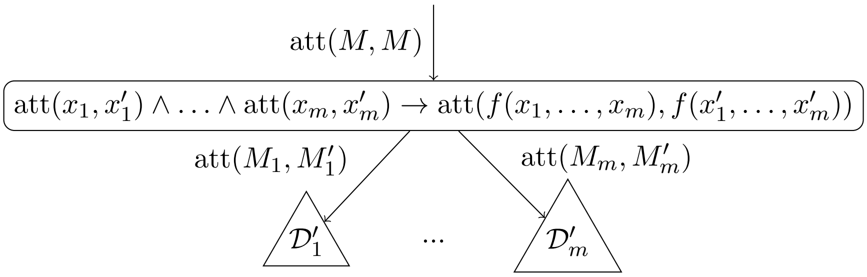
Proof. We prove this lemma by induction on the sizes of M, M', i.e. $|M| + |M'|$.

Base case $|M| + |M'| = 0$: Such a case is impossible hence the result trivially holds.

Inductive case $|M| + |M'| > 0$: Let us assume that $T \not\vdash \mathcal{D}$ (otherwise the result directly holds). Hence, $M = f(M_1, \ldots, M_m)$, $M' = f(M'_1, \ldots, M'_m)$ with $f \in \mathcal{F}_{data}$ and $\mathcal{D}$ is root by a node labelled by a rule R different from $\mathrm{att}(x_1, x'_1) \wedge \ldots \wedge \mathrm{att}(x_m, x'_M) \rightarrow \mathrm{att}(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m))$. Note that for all $i \in \{1, \ldots, m\}$, we can build the derivation $\mathcal{D}_i$ built as follows:


att(M_i, M'_i)
att$(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m)) \rightarrow$ att(x_i)
att(M, M')
$\mathcal{D}$

Notice that $T \vdash_w \mathcal{D}_i$ as the root node of $\mathcal{D}$ is now connected to the clause $\mathrm{att}(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m)) \rightarrow \mathrm{att}(x_i)$. Since $|M_i| + |M'_i| < |M| + |M'|$, we can apply on inductive hypothesis and deduce that there exists a derivation $\mathcal{D}'_i$ of $\mathrm{att}(M_i, M'_i)$ such that $T \vdash \mathcal{D}'_i$. We conclude by building the following derivation $\mathcal{D}'$:


$\mathrm{att}(M, M)$
$\mathrm{att}(x_1, x'_1) \wedge \ldots \wedge \mathrm{att}(x_m, x'_m) \rightarrow \mathrm{att}(f(x_1, \ldots, x_m), f(x'_1, \ldots, x'_m))$
$\mathrm{att}(M_1, M'_1)$
$\mathrm{att}(M_m, M'_m)$
$\mathcal{D}'_1$
...
$\mathcal{D}'_m$

□

Lemma 10 ([?]). *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 $\sigma'_1 \models \phi$.

Before proving Theorem 2, we state an invariant on the traces in $\mathsf{wrtrace}^2(\mathcal{C}_0)$.

Definition 26. *Let $\mathcal{C}_I = P_I, \mathcal{A}_I$ be an initial instrumented configuration. Let $T \in \mathsf{wrtrace}^2(\mathcal{C}_0)$. Let $\mathbb{C} = \mathbb{C}_\mathcal{A}(\mathcal{C}_I) \cup \mathbb{C}_\mathcal{P}(\mathcal{C}_I) \cup \mathbb{C}_e(T)$. Let τ be a step of T. We say $\mathcal{I}nv(T, \tau)$ holds if and only if $T[\tau] = \mathcal{P}, \mathcal{A}, \Delta$ and:*

1. *for all $(M, M') \in \mathcal{A}$, there exists a derivation $\mathcal{D}'$ of $\mathrm{att}(M, M')$ from $\mathbb{C}(\tau')$ such that $T \vdash \mathcal{D}'$; and*

for all $(P, P') \in \mathcal{P}$, there exist $Q, Q', \mathcal{H}, r, \sigma$ such that:

2. *$[\![|Q, Q'|]\!]\mathcal{H}r \subseteq \mathbb{C}_\mathcal{P}(\mathcal{C}_I)$; and*

3. *$P = Q\sigma$, $P' = Q'\sigma$; and*

4. *for all $F' \in \mathcal{H}\sigma$, there exists a derivation $\mathcal{D}'$ of F' from $\mathbb{C}$ such that $T \vdash \mathcal{D}'$.*

5. *for all formulae $\phi \in \mathcal{H}$, $\sigma \models \phi$.*

6. *either $r = \square$ or $r = \mathsf{diff}[o, o']$ with $\rightarrow \mathrm{ev}(repl(o\sigma), repl(o'\sigma)) \in \mathbb{C}_e(T)$*

7. *if $r = \mathsf{diff}[o, o']$ and P, P' starts with an input then $T[\tau - 1] \xrightarrow{(repl(o\sigma), repl(o'\sigma'))} T[\tau]$ by the rule* I^2-Repl .

Lemma 11. *Let $\mathcal{C}_I = (P_I, P'_I), \mathcal{A}_I$ be an initial instrumented configuration. Let $T \in \mathsf{wrtrace}^2($ $\mathcal{C}_I)$. Let $\mathbb{C} = \mathbb{C}_\mathcal{A}(\mathcal{C}_I) \cup \mathbb{C}_\mathcal{P}(\mathcal{C}_I) \cup \mathbb{C}_e(T)$. For all steps τ, $\mathcal{I}nv(T, \tau)$ holds.*

Proof. The proof is done by induction on τ.

Base case $\tau = 0$: $T[0] = \mathcal{C}_I = \mathcal{P}, \mathcal{A}, \Lambda$ with $\mathcal{P} = \{\!\{(P_I, P'_I)\}\!\}$, $\mathcal{A} = \mathcal{A}_I$ and $\Lambda = \emptyset$. By defining $Q = P_I$, $Q' = P'_I$, $\mathcal{H} = \top$ and $\sigma = id$, we directly obtain that item 3 of Definition 26 holds. Moreover, by definition, $\mathbb{C}_\mathcal{P}(\mathcal{C}_I) = [\![|P_I, P'_I|]\!]\top\square$ hence the item 2 holds. Since $\mathcal{H}\sigma = \top$, items 4 and 5 also hold. By definition of an initial instrumented semantics, $\mathcal{A}_I$ only contains the public names in P_I and P'_I. Moreover, we know that for all $a \in \mathcal{A}_I$, $\rightarrow \mathrm{att}(a, a)$ is a clause in $\mathbb{C}_{\mathcal{A}}(\mathcal{C}_I)$ (clauses RInit). Hence, we can build the a derivation of $\mathrm{att}(a,a)$ that is satisfied by T which allows us to conclude.

Inductive step $\tau > 0$: Since $\tau > 0$ and τ is a step of T, we know that there exists a step $\tau_0 = \tau - 1$ of T such that $T[\tau_0] \xrightarrow{\ell}_{i^2} T[\tau]$. By our inductive hypothesis, we deduce that $\mathcal{I}nv(T, \tau_0)$ holds. Let us denote $T[\tau_0] = \mathcal{P}_0, \mathcal{A}_0, \Lambda_0$ and $T[\tau] = \mathcal{P}, \mathcal{A}, \Lambda$.

We do a case analysis on the transition $T[\tau_0] \xrightarrow{\ell}_{i^2} T[\tau]$:

- Rule I^2-Nil: Trivial since $\mathcal{P} \subset \mathcal{P}_0$.

- Rule I^2-Par: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(P_1 \mid P_2, P_1' \mid P_2')\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P_1, P_1'), (P_2, P_2')\}\!\}$. By our inductive hypothesis, we know that there exist $Q_1, Q_1', Q_2, Q_2', \mathcal{H}, r, \sigma$ such that $(Q_1 \mid Q_2)\sigma = (P_1 \mid P_2)$, $(Q_1' \mid Q_2')\sigma = (P_1' \mid P_2')$ and $\llbracket | Q_1 \mid Q_2, Q_1' \mid Q_2' | \rrbracket \mathcal{H} r \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$. By definition, it implies that $\llbracket | Q_1, Q_1' | \rrbracket \mathcal{H} \square \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$ and $\llbracket | Q_2, Q_2' | \rrbracket \mathcal{H} \square \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$. Thus the result holds by associating $Q_1, Q_1', \mathcal{H}, \square, \sigma$ (resp. $Q_2, Q_2', \mathcal{H}, \square, \sigma$) to (P_1, P_1') (resp. (P_2, P_2')).

- Rule I^2-Repl: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}} P\}\!\} \cup \mathcal{M}, \{\!\{!^{\overline{o'}}_{\tilde{\mathsf{a}}'} P'\}\!\} \cup \mathcal{M}')\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P\sigma, P'\sigma'), (\{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}} P\}\!\} \cup \mathcal{M}, \{\!\{!^{\overline{o'}}_{\tilde{\mathsf{a}}'} P'\}\!\} \cup \mathcal{M}')\}\!\}$, $\Lambda = \Lambda_0 \cup \{(\overline{o}\sigma, \overline{o'}\sigma')\}$ with $dom(\sigma) = \tilde{a}$, $dom(\sigma') = \widetilde{a'}$, $img(\sigma) \cup img(\sigma') \subseteq \mathbb{N}$, $\overline{o}\sigma \notin proj_0(\Lambda_0)$, $\overline{o'}\sigma' \notin proj_1(\Lambda_0)$ and $\ell = (repl(\overline{o}\sigma), repl(\overline{o'}\sigma'))$.

  Recall that $\tilde{\mathsf{a}}$ and $\widetilde{\mathsf{a}'}$ are sequences of variables. Therefore, by our inductive hypothesis, we know that there exist $o_1, o_1', Q, Q', \mathcal{M}_1, \mathcal{M}_1'$, $\mathcal{H}$,σ_0 such that $\{\!\{!^{\overline{o}}_{\tilde{\mathsf{a}}} P\}\!\} \cup \mathcal{M} = (\{\!\{!^{\overline{o_1}}_{\tilde{\mathsf{a}}} Q\}\!\} \cup \mathcal{M}_1)\sigma_0$, $\{\!\{!^{\overline{o'}}_{\widetilde{\mathsf{a}'}} P'\}\!\} \cup \mathcal{M}' = (\{\!\{!^{\overline{o_1'}}_{\widetilde{\mathsf{a}'}} Q'\}\!\} \cup \mathcal{M}_1')\sigma_0$ and $\llbracket | \{\!\{!^{\overline{o_1}}_{\tilde{\mathsf{a}}} Q\}\!\} \cup \mathcal{M}_1, \{\!\{!^{\overline{o_1'}}_{\widetilde{\mathsf{a}'}} Q'\}\!\} \cup \mathcal{M}_1' | \rrbracket \mathcal{H} r \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

  By definition, $\llbracket | Q, Q' | \rrbracket \mathcal{H} \mathsf{diff}[\overline{o_1}, \overline{o_1'}] \subseteq \llbracket | \{\!\{!^{\overline{o_1}}_{\tilde{\mathsf{a}}} Q\}\!\} \cup \mathcal{M}_1, \{\!\{!^{\overline{o_1'}}_{\widetilde{\mathsf{a}'}} Q'\}\!\} \cup \mathcal{M}_1' | \rrbracket \mathcal{H} r$. Thus, by defining $\sigma_0' = \sigma_0\sigma\sigma'$, we deduce that $Q\sigma_0' = P\sigma$ and $Q'\sigma_0' = P'\sigma'$ hence item 2. Furthermore, as $\mathcal{H}\sigma_0 = \mathcal{H}\sigma_0'$, we deduce that items 4 and 5 hold.

  We showed that $\ell = (repl(\overline{o}\sigma), repl(\overline{o'}\sigma'))$ meaning that $\rightarrow \mathrm{ev}(repl(\overline{o}\sigma), repl(\overline{o'}\sigma'))$ is in $\mathbb{C}_e(T) \subseteq \mathbb{C}$. Since $\overline{o}\sigma = \overline{o_1}\sigma_0\sigma = \overline{o_1}\sigma_0'$ and $\overline{o'}\sigma = \overline{o_1'}\sigma_0\sigma' = \overline{o_1'}\sigma_0'$, we conclude that items 6 and 7 hold.

- Rule I^2-Comm: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{out}(N, M); P_1, \mathsf{out}(N', M'); P_1'), (\mathsf{in}^{\overline{o}}(N, x); P_2, \mathsf{in}^{\overline{o'}}(N', x'); P_2')\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P_1, P_1'), (P_2\{^M/_x\}, P_2'\{^{M'}/_{x'}\})\}\!\}$ and $\ell = (pre(\overline{o}, M), pre(\overline{o'}, M'))$. By our inductive hypothesis, we know that there exist $U_1, U_1', U_2, U_2', V, V'$, $Q_1, Q_1', Q_2, Q_2', \overline{o_1}, \overline{o_1'}$, $\mathcal{H}_1, \mathcal{H}_2, r_1, r_2, \sigma_1, \sigma_2$ such that

  - $(\mathsf{out}(N, M); P_1, \mathsf{out}(N', M'); P_1') = (\mathsf{out}(U_1, V); Q_1, \mathsf{out}(U_1', V'); Q_1')\sigma_1$,
  - $(\mathsf{in}^{\overline{o}}(N, x); P_2, \mathsf{in}^{\overline{o'}}(N', x'); P_2') = (\mathsf{in}^{\overline{o_1}}(U_2, x); Q_2, \mathsf{in}^{\overline{o_1'}}(U_2', x'); Q_2')\sigma_2$
  - $\llbracket | \mathsf{out}(U_1, V); Q_1, \mathsf{out}(U_1', V'); Q_1' | \rrbracket \mathcal{H}_1 r_1 \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$
  - $\llbracket | \mathsf{in}^{\overline{o_1}}(U_2, x); Q_2, \mathsf{in}^{\overline{o_1'}}(U_2', x'); Q_2' | \rrbracket \mathcal{H}_2 r_2 \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$

  By definition, we have:

  - $\llbracket | \mathsf{out}(U_1, V); Q_1, \mathsf{out}(U_1', V'); Q_1' | \rrbracket \mathcal{H}_1 r_1 = \llbracket | Q_1, Q_1' | \rrbracket (\mathcal{H}_1 \wedge G_1) \square \cup \{\mathcal{H}_1 \wedge G_1 \rightarrow \mathrm{msg}(U_1, V, U_1', V')\}$

- $[\![|\mathsf{in}^{\overline{o_1}}(U_2,x);Q_2,\mathsf{in}^{\overline{o'_1}}(U'_2,x');Q'_2|]\!]\mathcal{H}_2 r_2 = [\![|Q_2,Q'_2|]\!](\mathcal{H}_2 \wedge \mathrm{msg}(U_2,x,U'_2,x') \wedge \mathrm{ev}(\mathit{pre}(\overline{o_1},x),\mathit{pre}(\overline{o'_1},x')) \wedge G'_2)\Box \cup \{\mathcal{H}_2 \wedge G_2 \rightarrow \mathrm{input}(U_2,U'_2)\}$

where $G_1 = \top$ if $r_1 = \Box$ else $G_1 = \mathrm{ev}(\mathit{repl}(\mathit{proj}_0(r_1)),\mathit{repl}(\mathit{proj}_1(r_1)))$; and $G_2 = G'_2 = \top$ if $r_2 = \Box$ else $G_2 = \mathrm{ev}(\mathit{repl}(\mathit{proj}_0(r_2)),\mathit{repl}(\mathit{proj}_1(r_2)))$ and $G'_2 = \mathrm{ev}(\mathit{repl}_i(\mathit{proj}_0(r_1),x), \mathit{repl}_i(\mathit{proj}_1(r_2),x))$.

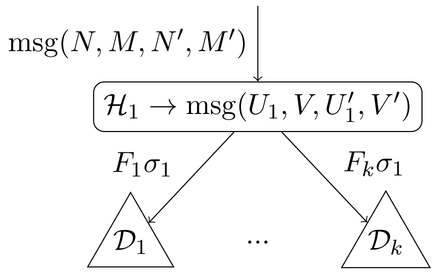
Let us define $\sigma'_2 = \sigma_2[x \mapsto M, x' \mapsto M']$. Thus, $P_2\{^M/_x\} = Q_2\sigma'_2$ and $P'_2\{^{M'}/_{x'}\} = Q'_2\sigma'_2$. Moreover, note that $U_1\sigma_1 = U'_1\sigma_2$ and $U'_1\sigma_1 = U'_1\sigma_2$. Therefore, items 2 and 3 hold by associating $Q_1, Q'_1, (\mathcal{H}_1 \wedge G_1), \Box, \sigma_1$ to (P_1, P'_1) and by associating $Q_2, Q'_2, (\mathcal{H}_2 \wedge \mathrm{msg}(U_2,x,U'_2,x') \wedge \mathrm{ev}(\mathit{pre}(\overline{o_1},x),\mathit{pre}(\overline{o'_1},x')) \wedge G'_2), \Box, \sigma'_2$ to $(P_2\{^M/_x\}, P'_2\{^{M'}/_{x'}\})$.

We now show the other items of Definition 26. Item 1 trivially hold since $\mathcal{A}_0 = \mathcal{A}$. Item 5 also trivially holds since no new formula has been added. Thus it remains to prove items 4 for $G_1, G'_2, \mathrm{msg}(U_2,x,U'_2,x)\sigma'_2$ and $\mathrm{ev}(\mathit{pre}(\overline{o_1},x),\mathit{pre}(\overline{o'_1},x'))\sigma'_2$.

Thanks to our inductive hypothesis, if $r_1 \neq \Box$ then we know that $\rightarrow \mathrm{ev}(\mathit{repl}(\mathit{proj}_0(r_1))\sigma_1, \mathit{repl}(\mathit{proj}_1(r_1))\sigma_1) \in \mathbb{C}_e(T) \subseteq \mathbb{C}_{\mathcal{P}}$. Moreover, if $r_2 \neq \Box$ then we know that $T[\tau_0 - 1] \xrightarrow{(\mathit{repl}(\mathit{proj}_0(r_2)\sigma_2, \mathit{repl}(\mathit{proj}_1(r_2)\sigma_2)}_{i^2} T[\tau_0]$. Hence, $T, \tau \vdash (\mathit{repl}_i(\mathit{proj}_0(r_2)\sigma_2, x\sigma'_2), \mathit{repl}_i(\mathit{proj}_1(r_2)\sigma_2, x'\sigma'_2)$. This allows us to deduce that $\rightarrow G'_2\sigma'_2$ is in $\mathbb{C}_e(T)$ and so in $\mathbb{C}$.

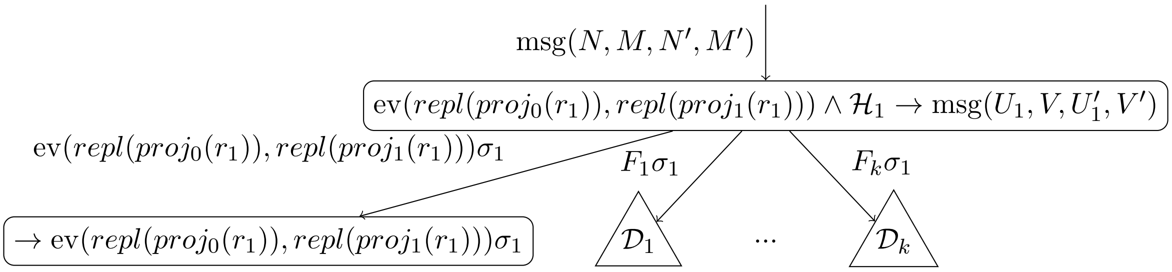
We already proved that $\overline{o} = \overline{o_1}\sigma_2$, $\overline{o'} = \overline{o'_1}\sigma_2$, $V\sigma_1 = M = x\sigma'_2$ and $V'\sigma_1 = M' = x'\sigma'_2$. But we have $T[\tau_0] \xrightarrow{\ell}_{wi^2} T[\tau]$ with $\ell = (\mathit{pre}(\overline{o},M),\mathit{pre}(\overline{o'},M'))$. Thus, $\rightarrow \mathrm{ev}(\mathit{pre}(\overline{o},M),\mathit{pre}(\overline{o'},M')) \in \mathbb{C}_e(T) \subseteq \mathbb{C}$. Hence, there exists a derivation $\mathcal{D}$ of $\mathrm{ev}(\mathit{pre}(\overline{o_1},x),\mathit{pre}(\overline{o'_1},x'))\sigma'_2$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}$.

Let us assume that $\mathcal{H}_1 = F_1 \wedge \ldots \wedge F_k \wedge \phi$. Since $\mathcal{I}nv(T,\tau_0)$ holds, $\sigma_1 \models \phi$ and there exist some derivations $\mathcal{D}_1,\ldots,\mathcal{D}_k$ of $F_1\sigma_1,\ldots,F_k\sigma_1$ respectively from $\mathbb{C}$ such that $T, \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. But the rule $\mathcal{H}_1 \rightarrow \mathrm{msg}(U_1,V,U'_1,V')$ is in $\mathbb{C}$. Since $\mathrm{msg}(U_1,V,U'_1,V')\sigma_1 = \mathrm{msg}(U_2,x,U'_2,x')\sigma'_2$, we can build the following derivation $\mathcal{D}$ of $\mathrm{msg}(N,M,N',M')$:



Since $T \vdash F_i$ for $1 \leq i \leq k$, we deduce that $T \vdash \mathcal{D}$ which allows us to conclude.

When $r_1 \neq \Box$, the derivation becomes


msg(N, M, N', M')
ev$(\mathit{repl}(\mathit{proj}_0(r_1)), \mathit{repl}(\mathit{proj}_1(r_1))) \wedge \mathcal{H}_1 \to$ msg(U_1, V, U'_1, V')
ev$(\mathit{repl}(\mathit{proj}_0(r_1)), \mathit{repl}(\mathit{proj}_1(r_1)))\sigma_1$
$F_1\sigma_1$
$F_k\sigma_1$
$\to$ ev$(\mathit{repl}(\mathit{proj}_0(r_1)), \mathit{repl}(\mathit{proj}_1(r_1)))\sigma_1$
$\mathcal{D}_1$
$\cdots$
$\mathcal{D}_k$

For the rest of the proof, we focus on the cases where $r = \Box$ as the arguments are the same as described here.

- Rule I^2-Let1: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P_1\ \mathsf{else}\ P_2, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'_1\ \mathsf{else}\ P'_2)\}\!\}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P_1\{^M/_x\}, P'_1\{^{M'}/_{x'}\})\}\!\}$ with $D \Downarrow M$ and $D' \Downarrow M'$.

  By our inductive hypothesis, we know that there exist $D_1, D'_1, Q_1, Q'_1, Q_2, Q'_2, \mathcal{H}, \sigma$ such that $(\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2)\sigma = (\mathsf{let}\ x = D\ \mathsf{in}\ P_1\ \mathsf{else}\ P_2)$, $(\mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2)\sigma = (\mathsf{let}\ x = D'\ \mathsf{in}\ P'_1\ \mathsf{else}\ P'_2)$ and $[\![|\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2, \mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2|]\!]\mathcal{H}\Box \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

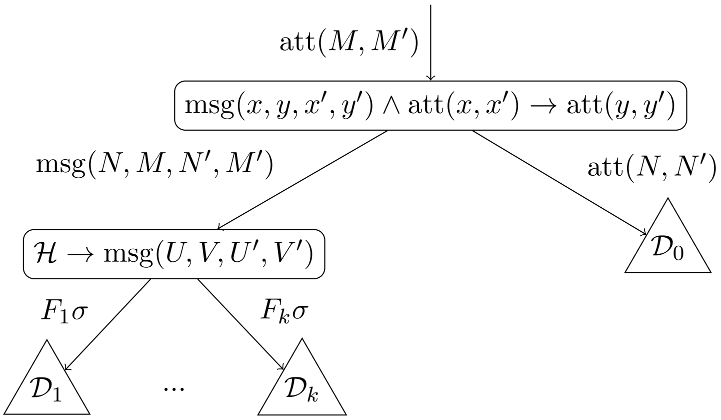
  Since $D = D_1\sigma$ and $D' = D'_1\sigma$, we can apply Lemma 10 to obtain that there exist $N, N', \sigma_1, \sigma'_1$, and ϕ such that $(D_1, D'_1) \Downarrow' ((N, N'), \sigma_1, \phi)$, $M = N\sigma'_1$, $M' = N'\sigma'_1$, $\sigma = (\sigma_1\sigma'_1)_{|dom(\sigma)}$ and $\sigma'_1 \models \phi$.

  By definition of $[\![|\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2, \mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2|]\!]\mathcal{H}\Box$, we have $[\![|Q_1\sigma_1\sigma''_1, Q'_1\sigma_1\sigma''_1|]\!](\mathcal{H}\sigma_1 \wedge \phi)\Box \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$ with $\sigma''_1 = \{^N/_x,^{N'}/_{x'}\}$. Note that $P_1\{^M/_x\} = (Q_1\sigma)\{^{N\sigma'_1}/_x\} = (Q_1\sigma_1\sigma'_1)\{^{N\sigma'_1}/_x\} = (Q_1\sigma_1\sigma''_1)\sigma'_1$. Similarly, $P'_1\{^{M'}/_{x'}\} = (Q'_1\sigma_1\sigma''_1)\sigma'_1$ implying that item 2 holds. The facts in $(\mathcal{H}\sigma_1)\sigma'_1$ are the same as in $\mathcal{H}\sigma$, hence item 4 holds. Finally, since $\sigma'_1 \models \phi$, we conclude that item 5 holds which allows us to conclude.

- Rule I^2-Let2: Similar to the previous case.

- Rule I^2-Out: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{out}(N, M); P, \mathsf{out}(N', M'); P')\}\!\}$, $N \in \mathcal{A}_0$, $\mathcal{A}_1 = \mathcal{A}_0 \cup \{(M, M')\}$, $(N, N') \in \mathcal{A}_0$ and $\mathcal{P} \cup \{\!\{(P, P')\}\!\}$. By our inductive hypothesis, we know that there exists $U, V, U', V', Q, Q', \mathcal{H}, \sigma$ such that $(\mathsf{out}(U, V); Q)\sigma = (\mathsf{out}(N, M); P)$, $(\mathsf{out}(U', V'); Q')\sigma = (\mathsf{out}(N', M'); P')$ and $[\![|\mathsf{out}(U, V); Q, \mathsf{out}(U', V'); Q'|]\!]\mathcal{H}\Box \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

  By definition $[\![|\mathsf{out}(U, V); Q, \mathsf{out}(U', V'); Q'|]\!]\mathcal{H}\Box = [\![|Q, Q'|]\!]\mathcal{H}\Box \cup \{\mathcal{H} \to \mathrm{msg}(U, V, U', V')\}$. But we know that $(N, N') \in \mathcal{A}_0$. Thus $\mathcal{I}nv(T, \tau')$ ensures that there exists a derivation $\mathcal{D}_0$ of (N, N') from $\mathbb{C}$ such that $T \vdash \mathcal{D}_0$. Moreover, item 4 also ensures that if we denote $\mathcal{H} = F_1 \wedge \ldots \wedge F_k \wedge \phi$ then $\sigma \models \phi$ and there exist some derivations $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of $F_1\sigma, \ldots, F_k\sigma$ respectively from $\mathbb{C}$ such that $T \vdash \mathcal{D}_j$ for all $j = 1 \ldots k$. Thus we can build a derivation $\mathcal{D}$ of att(M, M') from $\mathbb{C}$ as follows:


$\mathrm{att}(M,M')$
$\mathrm{msg}(x,y,x',y') \wedge \mathrm{att}(x,x') \to \mathrm{att}(y,y')$
$\mathrm{msg}(N,M,N',M')$
$\mathrm{att}(N,N')$
$\mathcal{H} \to \mathrm{msg}(U,V,U',V')$
$\mathcal{D}_0$
$F_1\sigma$
$F_k\sigma$
$\mathcal{D}_1$
...
$\mathcal{D}_k$

Note that we do not necessarily have that $T \vdash \mathcal{D}$ when M, M' have a data constructor function symbol at the root. However, since $T \vdash \mathcal{D}_i$ for $0 \le i \le k$, we deduce that $T \vdash_w \mathcal{D}$. We conclude by applying Lemma 9.

- Rule I^2-In: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\{(\mathsf{in}^{\overline{o}}(N,x);P, \mathsf{in}^{\overline{o'}}(N',x');P')\}\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\{(P\{^M/_x\}, P'\{^{M'}/_{x'}\})\}\!\}$ and $\ell = (pre(\overline{o},M), pre(\overline{o'},M'))$ with $(N,N'),(M,M') \in \mathcal{A}_0$.

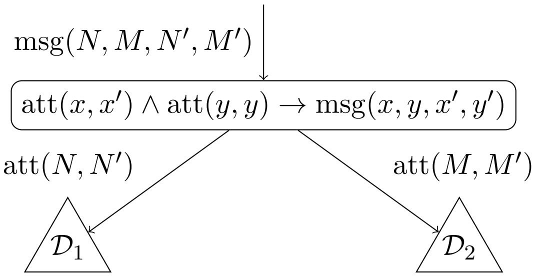
  By our inductive hypothesis, we know that there exist $U,U',Q,Q',\overline{o_1},\overline{o_1'},\mathcal{H},\sigma$ such that $(\mathsf{in}^{\overline{o_1}}(U,x);Q)\sigma = (\mathsf{in}^{\overline{o}}(N,x);P)$, $(\mathsf{in}^{\overline{o_1'}}(U',x');Q')\sigma = (\mathsf{in}^{\overline{o'}}(N',x');P')$ and $[\![|\mathsf{in}^{\overline{o_1}}(U,x); Q, \mathsf{in}^{\overline{o_1'}}(U',x');Q'|]\!]\mathcal{H}\square \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

  By definition, $[\![|\mathsf{in}^{\overline{o_1}}(U,x);Q,\mathsf{in}^{\overline{o_1'}}(U',x');Q'|]\!]\mathcal{H}\square = [\![|Q,Q|]\!](\mathcal{H} \wedge \mathrm{msg}(U,x,U',x') \wedge \mathrm{ev}(pre(\overline{o_1},x),pre(\overline{o_1'},x)))\square \cup \{\mathcal{H} \to \mathrm{input}(U,U')\}$.

  Let us define $\sigma' = \sigma[x \mapsto M, x' \mapsto M']$. We deduce that $P\{^M/_x\} = Q\sigma'$ and $P'\{^{M'}/_{x'}\} = Q'\sigma'$. We show that the result holds by assigning Q, Q', $(\mathcal{H} \wedge \mathrm{msg}(U,x,U',x') \wedge \mathrm{ev}(pre(\overline{o_1},x),pre(\overline{o_1'},x)))$ and σ' to $(P\{^M/_x\}, P'\{^{M'}/_{x'}\})$. To do so, it remains to define a derivation for $\mathrm{msg}(U,x,U',x')\sigma'$ and $\mathrm{ev}(pre(\overline{o_1},x),pre(\overline{o_1'},x)))\sigma'$.

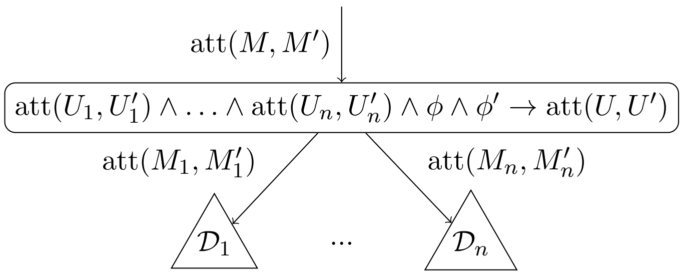
  We already know that $\ell = (pre(\overline{o},M),pre(\overline{o'},M'))$ and $T[\tau_0] \xrightarrow{\ell}_{wi^2} T[\tau]$ hence $\to \mathrm{ev}(pre(\overline{o},M),pre(\overline{o'},M'))$ is in $\mathbb{C}_e(T) \subseteq \mathbb{C}$. Since we proved that $\overline{o} = \overline{o_1}\sigma'$, $\overline{o'} = \overline{o_1'}\sigma'$, $x\sigma' = M$ and $x'\sigma' = M'$, we deduce that there exists a derivation $\mathcal{D}$ of $\mathrm{ev}(pre(\overline{o_1},x),pre(\overline{o_1'},x)))\sigma'$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}$.

  We know that $\mathcal{I}nv(T,\tau_0)$ holds and in particular item 1. Thus, $(M,M'),(N,N') \in \mathcal{A}_0$ implies that there exist some derivations $\mathcal{D}_1,\mathcal{D}_2$ of $\mathrm{att}(M,M'),\mathrm{att}(N,N')$ respectively from $\mathbb{C}$ such that $T \vdash \mathcal{D}_1$ and $T \vdash \mathcal{D}_2$. As we have already proved that $U\sigma' = N$ and $U'\sigma' = N'$, we can build a derivation $\mathcal{D}$ of $\mathrm{msg}(U,x,U',x')\sigma'$ from $\mathbb{C}$ as follows:


$\mathrm{msg}(N,M,N',M')$
$\mathrm{att}(x,x') \wedge \mathrm{att}(y,y) \to \mathrm{msg}(x,y,x',y')$
$\mathrm{att}(N,N')$
$\mathrm{att}(M,M')$
$\mathcal{D}_1$
$\mathcal{D}_2$

- Rule I^2-App: In such a case $\mathcal{A} = \mathcal{A}_0 \cup \{(M, M')\}$ where $(M_1, M'_1), \dots, (M_n, M'_n) \in \mathcal{A}_0$, $f \in \mathcal{F}_d \cup \mathcal{F}_c$ and $f(M_1, \dots, M_n) \Downarrow M$ and $f(M'_1, \dots, M'_n) \Downarrow M'$. Note that $M_1, M'_1, \dots, M_m, M'_m$ are terms. Thus by definition of $\Downarrow$, $f(M_1, \dots, M_n) \Downarrow M$ implies that there exist $f(U_1, \dots, U_n) \to U \mid\mid \phi \in \mathsf{def}(f)$ and a substitution σ such that $U_i\sigma = M_i$ for all $i \in \{1, \dots, n\}$, $M = U\sigma$ and $\sigma \models \phi$. Similarly, $f(M'_1, \dots, M'_n) \Downarrow M'$ implies that there exist $f(U'_1, \dots, U'_m) \to U' \mid\mid \phi' \in \mathsf{def}(f)$ and a substitution σ' such that $U'_i\sigma' = M'_i$ for all $i \in \{1, \dots, n\}$, $M' = U'\sigma'$ and $\sigma' \models \phi'$. By considering the variables of $f(U_1, \dots, U_n)$ and $f(U'_1, \dots, U'_n)$ distinct, we deduce that there exists a substitution σ_0 such that $(U_i\sigma_0, U'_i\sigma_0) = (M_i, M'_i)$ for all $i \in \{1, \dots, n\}$, $(M, M') = (U, U')\sigma_0$ and $\sigma_0 \models \phi \wedge \phi'$.

  Moreover, since $\mathcal{I}nv(T, \tau_0)$ holds and $(M_1, M'_1), \dots, (M_n, M'_n) \in \mathcal{A}_0$, we know that there exist some derivations $\mathcal{D}_1, \dots, \mathcal{D}_n$ of $\mathrm{att}(M_1, M'_1)$, …, $\mathrm{att}(M_n, M'_n)$ respectively from $\mathbb{C}$ such that $T \vdash \mathcal{D}_j$ for all $j = 1 \dots n$. Therefore, we can build a derivation $\mathcal{D}$ of $\mathrm{att}(M, M')$ from $\mathbb{C}$ as follows:

  
  $\mathrm{att}(M, M')$
$\mathrm{att}(U_1, U'_1) \wedge \dots \wedge \mathrm{att}(U_n, U'_n) \wedge \phi \wedge \phi' \to \mathrm{att}(U, U')$
$\mathrm{att}(M_1, M'_1)$
$\mathrm{att}(M_n, M'_n)$
$\mathcal{D}_1$
...
$\mathcal{D}_n$

  Note that we do not necessarily have that $T \vdash \mathcal{D}$ when M, M' have a data constructor function symbol at the root. However, since $T \vdash \mathcal{D}_i$ for $0 \leq i \leq n$, we deduce that $T \vdash_w \mathcal{D}$. We conclude by applying Lemma 9.

- Rule I^2-Gen: Direct by application of the clause RGen.

- Rule I^2-Event: In such a case, $\mathcal{P}_0 = \mathcal{P}' \cup \{\!\!\{(\mathsf{event}(ev); P, \mathsf{event}(ev'); P'\}\!\!\}$, $\mathcal{P} = \mathcal{P}' \cup \{\!\!\{(P, P')\}\!\!\}$ and $\ell = (ev, ev')$. By our inductive hypothesis, there exist $ev_1, ev'_1, Q, Q', \mathcal{H}, \sigma$ such that $(\mathsf{event}(ev_1); Q)\sigma = (\mathsf{event}(ev); P)$, $(\mathsf{event}(ev'_1); Q')\sigma = (\mathsf{event}(ev'); P')$ and $[\![|\mathsf{event}(ev_1); Q, \mathsf{event}(ev'_1); Q'|]\!]\mathcal{H}\square \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

  By definition, $[\![|\mathsf{event}(ev_1); Q, \mathsf{event}(ev'_1); Q'|]\!]\mathcal{H}\square = [\![|Q, Q'|]\!](\mathcal{H} \wedge \mathrm{ev}(ev_1, ev'_1))\square$. Since $ev_1\sigma = ev$, $ev'_1\sigma = ev'$, $\ell = (ev, ev')$ and $T[\tau_0] \xrightarrow{\ell}_{wi^2} T[\tau]$, we deduce that $\to \mathrm{ev}(ev, ev')$ is in $\mathbb{C}_e(T)$ and so in $\mathbb{C}$. Hence, we deduce that there exists a derivation $\mathcal{D}$ of $\mathrm{ev}(ev_1, ev'_1)\sigma$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}$, which allows us to conclude. □

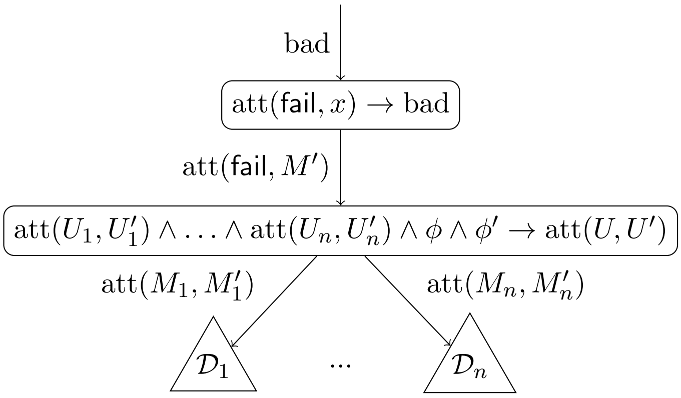
Theorem 2 (Soundness Initial Clauses)**.** *Let $\mathcal{C}$ be an initial instrumented biconfiguration. For all $T \in \mathsf{wrtrace}^2(\mathcal{C})$, if T does not converge then there exists a derivation of* bad *from $\mathbb{C}_{\mathcal{P}}(\mathcal{C}) \cup \mathbb{C}_e(T)$.*

Proof. Since T does not converge, we deduce that there exists a step τ such that $T[\tau]$ does not converge. By Lemma 11, we know that $\mathcal{I}nv(T, \tau)$ holds. Let us denote $T[\tau] = \mathcal{P}, \mathcal{A}, \Lambda$. We do a case analysis on why $T[\tau]$ does not converge by looking at Definition 9.

- Case $(M_1, M'_1), \ldots, (M_n, M'_n) \in \mathcal{A}$, $f \in \mathcal{F}_d$ and $f(M_1, \ldots, M_n) \Downarrow \mathsf{fail}$ but $f(M'_1, \ldots, M'_n) \Downarrow M'$ for some M' (the case $f(M_1, \ldots, M_n) \Downarrow M$ but $f(M'_1, \ldots, M'_n) \Downarrow \mathsf{fail}$ is symmetrical): Thanks to $\mathcal{I}nv(T, \tau)$, we know that there exists derivations $\mathcal{D}_1, \ldots, \mathcal{D}_n$ of $(M_1, M'_1), \ldots, (M_n, M'_n)$ respectively such that $T \vdash \mathcal{D}_i$ for $1 \leq i \leq n$.

  Note that $M_1, M'_1, \ldots, M_m, M'_m$ are terms. Thus by definition of $\Downarrow$, $f(M_1, \ldots, M_n) \Downarrow \mathsf{fail}$ implies that there exist $f(U_1, \ldots, U_n) \to U \mid\mid \phi \in \mathsf{def}(f)$ and a substitution σ such that $U_i\sigma = M_i$ for all $i \in \{1, \ldots, n\}$, $\mathsf{fail} = U\sigma$ and $\sigma \models \phi$. Similarly, $f(M'_1, \ldots, M'_n) \Downarrow M'$ 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 $i \in \{1, \ldots, n\}$, $M' = U'\sigma'$ and $\sigma' \models \phi'$. By considering the variables of $f(U_1, \ldots, U_n)$ and $f(U'_1, \ldots, U'_n)$ distinct, we deduce that there exists a substitution σ_0 such that $(U_i\sigma_0, U'_i\sigma_0) = (M_i, M'_i)$ for all $i \in \{1, \ldots, n\}$, $(M, M') = (U, U')\sigma_0$ and $\sigma_0 \models \phi \wedge \phi'$.

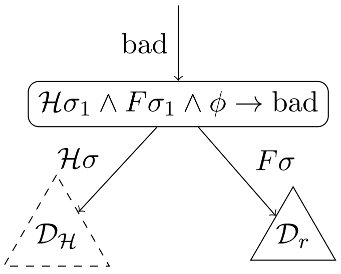
  Therefore, combining the appropriate clause from (Rf) and the clause (RIBad2), we can build the following derivation:


bad
$\mathrm{att}(\mathsf{fail}, x) \to \mathrm{bad}$
$\mathrm{att}(\mathsf{fail}, M')$
$\mathrm{att}(U_1, U'_1) \wedge \ldots \wedge \mathrm{att}(U_n, U'_n) \wedge \phi \wedge \phi' \to \mathrm{att}(U, U')$
$\mathrm{att}(M_1, M'_1)$
$\mathrm{att}(M_n, M'_n)$
$\mathcal{D}_1$
...
$\mathcal{D}_n$

- Case $\mathcal{P} = \mathcal{P}' \cup \{\!\{(\mathsf{let}\ x = D\ \mathsf{in}\ P_1\ \mathsf{else}\ P_2, \mathsf{let}\ x' = D'\ \mathsf{in}\ P'_1\ \mathsf{else}\ P'_2)\}\!\}$ and $D \Downarrow \mathsf{fail}$ but $D' \Downarrow M'$ for some M' (the case $D \Downarrow M$ and $D' \Downarrow \mathsf{fail}$ is symmetrical): Thanks to $\mathcal{I}nv(T, \tau)$, we know that there exist $D_1, D'_1, Q_1, Q_2, Q'_1, Q'_2$, $\mathcal{H}$, r and σ such that $(\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2)\sigma = (\mathsf{let}\ x = D\ \mathsf{in}\ P_1\ \mathsf{else}\ P_2)$, $(\mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2)\sigma = (\mathsf{let}\ x = D'\ \mathsf{in}\ P'_1\ \mathsf{else}\ P'_2)$ and $[\![|\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2, \mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2|]\!]\mathcal{H}\square \subseteq \mathbb{C}_{\mathcal{P}}(\mathcal{C}_I)$.

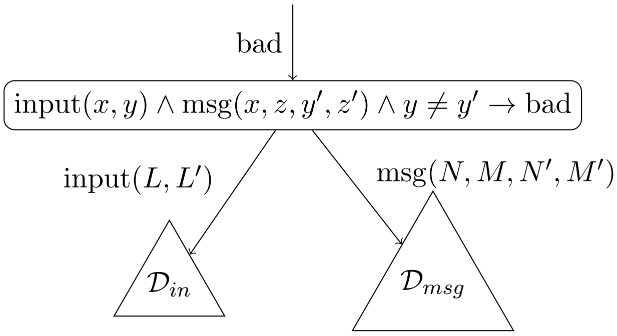
  Since $D = D_1\sigma$ and $D' = D'_1\sigma$, we can apply Lemma 10 to obtain that there exist $U, U', \sigma_1, \sigma'_1$, and ϕ such that $(D_1, D'_1) \Downarrow' ((U, U'), \sigma_1, \phi)$, $\mathsf{fail} = U\sigma'_1$, $M' = U'\sigma'_1$, $\sigma = (\sigma_1\sigma'_1)_{|dom(\sigma)}$ and $\sigma'_1 \models \phi$.

  By definition of $[\![|\mathsf{let}\ x = D_1\ \mathsf{in}\ Q_1\ \mathsf{else}\ Q_2, \mathsf{let}\ x = D'_1\ \mathsf{in}\ Q'_1\ \mathsf{else}\ Q'_2|]\!]\mathcal{H}\square$, we have that the following clause is in $\mathbb{C}$:
$$\mathcal{H}\sigma_1 \wedge F\sigma_1 \wedge \phi \to \mathrm{bad}$$
  with $F = \top$ if $r = \square$ else $F = \mathrm{ev}(\mathit{repl}(\mathit{proj}_0(r)), \mathit{repl}(\mathit{proj}_1(r)))$. Moreover, item 4 of Definition 26 indicates that there exists a conjunction of derivation $\mathcal{D}_{\mathcal{H}}$ of $\mathcal{H}\sigma$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}_{\mathcal{H}}$. Finally, item 6 of Definition 26 indicate that if $r \neq \square$ then $\to \mathrm{ev}(\mathit{repl}(\mathit{proj}_0(r)), \mathit{repl}(\mathit{proj}_1(r)))\sigma$ is in $\mathbb{C}$. Hence, there exists a derivation $\mathcal{D}_r$ of F (by abuse of notation, we consider $\mathcal{D}_r$ empty when $F = \top$) such that $T \vdash F$. This allows us to build the following derivation

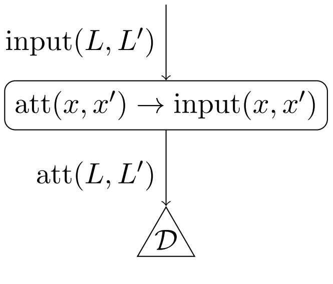

bad
$\mathcal{H}\sigma_1 \wedge F\sigma_1 \wedge \phi \rightarrow \text{bad}$
$\mathcal{H}\sigma$
$F\sigma$
$\mathcal{D}_{\mathcal{H}}$
$\mathcal{D}_r$

- One communication succeeds on one side but not on the other: In such a case, one the following case holds (i) $\mathcal{P} = \mathcal{P}'\cup\{\!\{(\mathsf{out}(N,M);P,\mathsf{out}(N',M');P'),(\mathsf{in}^{\overline{o}}(L,x);Q,\mathsf{in}^{\overline{o'}}(L',x');Q')\}\!\}$, (ii) $(L,L') \in \mathcal{A}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(\mathsf{out}(N,M);P;\mathsf{out}(N',M');P')\}\!\}$ (iii) $(N,N') \in \mathcal{A}$ and $\mathcal{P} = \mathcal{P}' \cup \{\!\{(\mathsf{in}^{\overline{o}}(L,x);P,\mathsf{in}^{\overline{o'}}(L',x');P')\}\!\}$ Additionally, $N = L$ and $N' \neq L'$ (the case $N \neq L$ and $N' = L'$ is symmetrical).

  In all three cases, we will show that there exist a derivation $\mathcal{D}_{in}$ of input(L,L') from $\mathbb{C}$; and a derivation $\mathcal{D}_{msg}$ of msg(N,M,N',M') from $\mathbb{C}$ from some M,M'. Once these two derivations obtained, we can generate the following derivation to conclude:


bad
$\text{input}(x,y) \wedge \text{msg}(x,z,y',z') \wedge y \neq y' \rightarrow \text{bad}$
input(L,L')
msg(N,M,N',M')
$\mathcal{D}_{in}$
$\mathcal{D}_{msg}$

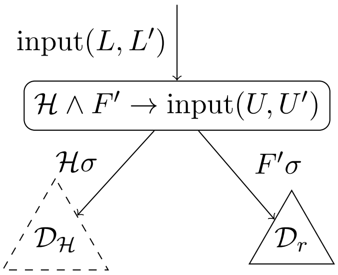
  When $(L,L') \in \mathcal{A}$, we know from item 1 of $\mathcal{I}nv(T,\tau)$ that there exists a derivation $\mathcal{D}$ of att(L,L'). Hence, we can build the derivation $\mathcal{D}_{in}$:



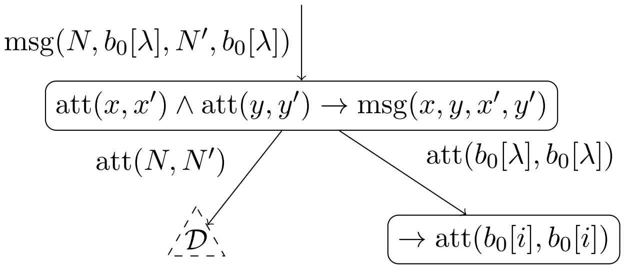
  When $(\mathsf{in}^{\overline{o}}(L,x);P,\mathsf{in}^{\overline{o'}}(L',x');P')$ is in the multiset $\mathcal{P}$, we know from $\mathcal{I}nv(T,\tau)$ that there exist $U,U',o_1,o_1',Q,Q',\mathcal{H},r,\sigma$ such that $\mathsf{in}^{\overline{o}}(L,x);P = (\mathsf{in}^{\overline{o_1}}(U,x);Q)\sigma$, $\mathsf{in}^{\overline{o'}}(L',x');P' = (\mathsf{in}^{\overline{o_1'}}(U',x');Q')\sigma$ and $[\![|\mathsf{in}^{\overline{o_1}}(U,x);Q,\mathsf{in}^{\overline{o_1'}}(U',x');Q'|]\!]\mathcal{H}r \subseteq \mathbb{C}$. In particular, the clause $\mathcal{H}\wedge F' \rightarrow \text{input}(U,U')$ is in $\mathbb{C}$ where $F' = \top$ if $r = \square$ or $F' = \text{ev}(\mathit{repl}(\mathit{proj}_0(r)),\mathit{repl}(\mathit{proj}_1(r)))$.

  From item 6, we additionally know that if $r \neq \square$ then $\rightarrow \text{ev}(\mathit{repl}(\mathit{proj}_0(r)),\mathit{repl}(\mathit{proj}_1(r)))\sigma$ is in $\mathbb{C}$. Hence, there exists a derivation $\mathcal{D}_r$ of $F'\sigma$ (by abuse of notation, we consider $\mathcal{D}_r$

empty when $F = \top$) such that $T \vdash F$. Finally, from item 4 and 5, we know that there exists a conjunction of derivation $\mathcal{D}_{\mathcal{H}}$ of $\mathcal{H}\sigma$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}_{\mathcal{H}}$. This allows us to build the following derivation $\mathcal{D}_{in}$:

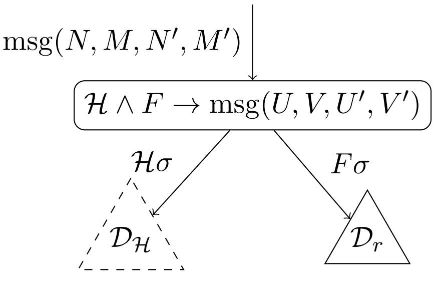

input(L, L')
$\mathcal{H} \wedge F' \rightarrow$ input(U, U')
$\mathcal{H}\sigma$
$F'\sigma$
$\mathcal{D}_{\mathcal{H}}$
$\mathcal{D}_r$

Let us now look at the derivation $\mathcal{D}_{msg}$. When $(N, N') \in \mathcal{A}$, we know from item 1 of $\mathcal{I}nv(T, \tau)$ that there exists a derivation $\mathcal{D}$ of att(N, N'). Hence, by taking any λ, we can build the derivation $\mathcal{D}_{msg}$:


msg$(N, b_0[\lambda], N', b_0[\lambda])$
att$(x, x') \wedge$ att$(y, y') \rightarrow$ msg(x, y, x', y')
att(N, N')
att$(b_0[\lambda], b_0[\lambda])$
$\mathcal{D}$
$\rightarrow$ att$(b_0[i], b_0[i])$

Finally when $(\mathsf{out}(N, M); P, \mathsf{out}(N', M'); P')$ is in the multiset $\mathcal{P}$, we know from $\mathcal{I}nv(T, \tau)$ that there exist $U, U', V, V', Q, Q', \mathcal{H}, r, \sigma$ such that $\mathsf{out}(N, M); P = (\mathsf{out}(U, V); Q)\sigma$, $\mathsf{out}(N', M'); P' = (\mathsf{out}(U', V'); Q')\sigma$ and $[\![|\mathsf{out}(U, V); Q, \mathsf{out}(U', V'); Q'|]\!]\mathcal{H}r \subseteq \mathbb{C}$. In particular, the clause $\mathcal{H} \wedge F \rightarrow$ msg(U, V, U', V') is in $\mathbb{C}$ where $F = \top$ if $r = \square$ or $F = \text{ev}(repl(proj_0(r)), repl(proj_1(r)))$.

From item 6, we additionally know that if $r \neq \square$ then $\rightarrow$ ev$(repl(proj_0(r)), repl($ $proj_1(r)))\sigma$ is in $\mathbb{C}$. Hence, there exists a derivation $\mathcal{D}_r$ of $F\sigma$ (by abuse of notation, we consider $\mathcal{D}_r$ empty when $F = \top$) such that $T \vdash F$. Finally, from item 4 and 5, we know that there exists a conjunction of derivation $\mathcal{D}_{\mathcal{H}}$ of $\mathcal{H}\sigma$ from $\mathbb{C}$ such that $T \vdash \mathcal{D}_{\mathcal{H}}$. This allows us to build the following derivation $\mathcal{D}_{msg}$:


msg(N, M, N', M')
$\mathcal{H} \wedge F \rightarrow$ msg(U, V, U', V')
$\mathcal{H}\sigma$
$F\sigma$
$\mathcal{D}_{\mathcal{H}}$
$\mathcal{D}_r$

This conclude with our proof that bad is derivable. □

D Proofs of semantics conditions (Theorems 3 to 5)

By abuse of notation, when $\mathcal{C} \rightarrow_{i^2} \mathcal{C}'$ by a rule without label, we may write $\mathcal{C} \xrightarrow{(\varepsilon,\varepsilon)}_{i^2} \mathcal{C}'$ where ε stands for the empty word.

Given a trace $T = \mathcal{C}_0 \xrightarrow{\ell_1}_i \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$, for $k \in \{0, \dots, n\}$, we denote by $T_{|k}$ the trace $\mathcal{C}_0 \xrightarrow{\ell_1}_i \dots \xrightarrow{\ell_k}_i \mathcal{C}_k$. We use a similar notation for bitraces.

Given a bitrace T, a mapping ρ and $i \in \{0, 1\}$, we denote by $\mathcal{P}_i(\rho, T)$ the predicate that holds when for all $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, if $T \vdash^2 (ev_0, ev_1)$ and $\mathsf{orepl}(ev_{1-i}) = o_{1-i}[\tilde{\mathsf{a}}_{1-i}]$ then $ev_i \in dom(\rho)$ and $\rho(ev_i) = o_{1-i}[\tilde{\mathsf{a}}_{1-i|\lambda}]$.

Lemma 12. *Let $\mathcal{C}_I = (\{\!\!\{(P_0, P_1)\}\!\!\}, \mathcal{A}, \emptyset)$ be an initial instrumented biconfiguration. Let $\mathbb{C}_{sat} = \mathsf{saturate}(\mathbb{C}_{\mathcal{P}}(\mathcal{C}_I))$. Let $i \in \{0, 1\}$. Let $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$ with $|T| = n$. Let ρ be a matching mapping from P_i to P_{1-i}. Assume that:*

- *for all $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T \vdash ev$ implies $ev \in dom(\rho)$*
- *for all $C = (H \rightarrow \text{bad}) \in \mathbb{C}_{sat}$, $\rho \vdash_{cs} \mathsf{falsify}_i(C)$*

For all $k_0 \in \{0, \dots, n\}$, for all $T_{k_0} \in \mathsf{wtrace}^2(\mathcal{C}_I)$, if $proj_i(T_{k_0}) = T_{|k_0}$ and $\mathcal{P}_i(\rho, T_{k_0})$ then there exists $T_n \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $proj_i(T_n) = T$, $T_{n|k_0} = T_{k_0}$ and $\mathcal{P}_i(\rho, T_n)$.

Proof. We prove the property for $i = 0$, the case $i = 1$ being symmetrical. Let us assume that $T = \mathcal{C}_0 \xrightarrow{\ell_1}_i \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$ where $\mathcal{C}_0 = proj_i(\mathcal{C}_I)$. We will prove by induction on $k \in \{k_0, \dots, n\}$ that there exists a bitrace $T_k \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $proj_0(T_k) = T_{|k}$, $T_{k|k_0} = T_{k_0}$ and $\mathcal{P}_0(\rho, T_k)$. Once this property proved, we conclude by taking $k = n$.

Base case $k = k_0$: In this case, We take $T_k = T_{k_0}$. We have by hypothesis that $proj_0(T_k) = T_{|k}$ and $\mathcal{P}_0(\rho, T_k)$. Hence the result holds.

Inductive step $k > k_0$: In this case, by our inductive hypothesis, we know that there exists a bitrace $T_{k-1} \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $proj_0(T_{k-1}) = T_{|k-1}$, $T_{k-1|k_0} = T_{k_0}$ and $\mathcal{P}_0(\rho, T_{k-1})$. By Lemma 5, we deduce that T_{k-1} converges.

We do a case analysis on the transition rule $\mathcal{C}_{k-1} \xrightarrow{\ell_k}_{wi} \mathcal{C}_k$:

- *Rule* I-Repl*:* In such a case, $\ell_k \in \mathsf{Ev}_!$ and $T, k \vdash \ell_k$.Thus, by hypothesis, $\ell_k \in dom(\rho)$. By definition of the transition, $\ell_k = repl(o[\tilde{\mathsf{a}}])$ for some $o \in \mathcal{N}_{!o}$ and $\tilde{\mathsf{a}}$. Moreover, by definition of a matching mapping, $\ell_k \in dom(\rho)$ implies that $\rho(\ell_k) = o'[\tilde{\mathsf{b}}]$ with $(o, o') \in \mathrm{pm}(\mathcal{C}_I)$ and $\mathsf{ar}^{\lambda}_{\mathcal{C}_I}(o') = |\tilde{\mathsf{b}}|$. Let us denote $ev_0 = repl(o[\tilde{\mathsf{a}}])$. We have $ev_0 \in \mathsf{Ev}_!$ with $\mathsf{orepl}(ev_0) = o[\tilde{\mathsf{a}}]$.

  We now show how to extend T_{k-1} into a bitrace T_k. We aim to apply Lemma 4. Let us take $\lambda_1, \dots, \lambda_r = \tilde{\mathsf{b}}_{|\lambda}$. Consider $(o'_0[\tilde{\mathsf{a}}'_0], o'_1[\tilde{\mathsf{a}}'_1]) \in \Lambda(\mathcal{C}_{k-1})$. By definition of the semantics rules, $T_{|k-1} \vdash^2 (repl(o'_0[\tilde{\mathsf{a}}'_0]), repl(o'_1[\tilde{\mathsf{a}}'_1]))$ with $o'_0, o'_1 \in \mathcal{N}_!$.

  Let us denote by ev'_0, ev'_1 the events $repl(o'_0[\tilde{\mathsf{a}}'_0]), repl(o'_1[\tilde{\mathsf{a}}'_1])$ respectively. Hence $ev'_0, ev'_1 \in \mathsf{Ev}_!$ with $\mathsf{orepl}(ev'_0) = o'_0[\tilde{\mathsf{a}}'_0]$ and $\mathsf{orepl}(ev'_1) = o'_1[\tilde{\mathsf{a}}'_1]$. Thanks to our inductive hypothesis, we obtain $\rho(ev'_0) = o'_1[\tilde{\mathsf{a}}'_{1|\lambda}]$.

  As $\rho(ev'_0) = o'_1[\tilde{\mathsf{a}}'_{1|\lambda}]$ and $\rho(ev_0) = o'[\tilde{\mathsf{b}}]$, we deduce by definition of a matching mapping that $repl(o'_0[\tilde{\mathsf{a}}'_0]) \prec_{\mathcal{C}_I} o[\tilde{\mathsf{a}}]$ implies $o'_1[\tilde{\mathsf{a}}'_{1|\lambda}] \prec_{\mathcal{C}_I} o'[\tilde{\mathsf{b}}]$.

We now need to prove that $o'_1[\tilde{\mathsf{a}}'_{1|\lambda}] \neq o'[\tilde{\mathsf{b}}]$. Let us assume by contradiction that $o'_1[\tilde{\mathsf{a}}'_{1|\lambda}] = o'[\tilde{\mathsf{b}}]$. Hence, $\rho(ev'_0) = \rho(ev_0)$. By definition of matching mapping, it implies that $o'_0 = o$ and $\tilde{\mathsf{a}}'_{0|\lambda} = \tilde{\mathsf{a}}_{|\lambda}$. By Lemma 2, we deduce that $\tilde{\mathsf{a}}'_0 = \tilde{\mathsf{a}}$ and so $o[\tilde{\mathsf{a}}] \in \Lambda(\mathcal{C}_{k-1})$ which is in contradiction with the application of the rule $\mathcal{C}_{k-1} \xrightarrow{\ell_k}_i \mathcal{C}_k$. Therefore, $o'_1[\tilde{\mathsf{a}}'_{1|\lambda}] \neq o'[\tilde{\mathsf{b}}]$.

We can therefore apply Lemma 4, to obtain that $T_{k-1}[k-1] \xrightarrow{(ev_0,ev_1)}_{wi^2} \mathcal{C}'_k$ with $ev_1 = \mathit{repl}(o'[\tilde{\mathsf{a}}'])$, $\mathit{proj}_0(\mathcal{C}'_k) = \mathcal{C}_k$ and $\tilde{\mathsf{a}}'_{|\lambda} = \tilde{\mathsf{b}}$. As $\rho(ev_0) = o'[\tilde{\mathsf{b}}]$, we conclude by taking $T_k = T_{k-1} \xrightarrow{(ev_0,ev_1)}_{wi^2} \mathcal{C}'_k$.

- *Rule* I-In *or* I-Comm*:* By Lemma 1, we directly obtain that there exists a biconfiguration $\mathcal{C}'_k$ such that $T_{k-1} \xrightarrow{(\ell_k,\ell'_k)}_{wi^2} \mathcal{C}'_k$ and $\mathit{proj}_0(\mathcal{C}'_k) = \mathcal{C}_k$. Let us denote $T_k = T_{k-1} \xrightarrow{(\ell_k,\ell'_k)}_{wi^2} \mathcal{C}'_k$. Hence $\mathit{proj}_0(T_k) = T_{|k}$.

  Let us first assume that $\mathcal{C}_{k-2} \not\xrightarrow{\ell_{k-1}\cdot\ell_k}_{wi} \mathcal{C}_k$. In such a case, for all $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T_k \vdash^2 (ev_0, ev_1)$ implies that $T_{k-1} \vdash^2 (ev_0, ev_1)$. Hence we can directly conclude by our inductive hypothesis.

  Let us now assume that $\mathcal{C}_{k-2} \xrightarrow{\ell_{k-1}\cdot\ell_k}_{wi} \mathcal{C}_k$. In such a case, $\ell_{k-1} = \mathit{repl}(o_0[\tilde{\mathsf{a}}_0])$ and $\ell_k = \mathit{pre}(o'_0[\tilde{\mathsf{a}}_{0|\lambda}], M_0)$ for some $o_0, o'_0, \tilde{\mathsf{a}}, M_0$. Moreover, as $\mathit{proj}_0(T_k) = T_{|k}$, we deduce that $T_k, k-1 \vdash^2 (\mathit{repl}(o_0[\tilde{\mathsf{a}}_0]), \mathit{repl}(o_1[\tilde{\mathsf{a}}_1])$ and $T_k, k \vdash^2 (\mathit{repl}_i(o_0[\tilde{\mathsf{a}}_0], M_0), \mathit{repl}_i(o_1[\tilde{\mathsf{a}}_1], M_1))$ for some $o_1, \tilde{\mathsf{a}}_1, M_1$.

  Note that for all $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T_k \vdash^2 (ev_0, ev_1)$ implies that either $T_{k-1} \vdash^2 (ev_0, ev_1)$ or $ev_0 = \mathit{repl}_i(o_0[\tilde{\mathsf{a}}_0], M_0)$ and $ev_1 = \mathit{repl}_i(o_1[\tilde{\mathsf{a}}_1], M_1)$. We need to show that $ev_0 \in \mathit{dom}(\rho)$ and $\rho(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$.

  As $\mathcal{C}_{k-2} \not\xrightarrow{\ell_{k-1}\cdot\ell_k}_{wi} \mathcal{C}_k$, we directly have that $T_{|k} \vdash \mathit{repl}_i(o_0[\tilde{\mathsf{a}}_0], M_0)$. As $T_{|k}$ is a prefix of T, we have that $T \vdash \mathit{repl}_i(o_0[\tilde{\mathsf{a}}_0], M_0)$ and so by hypothesis $ev_0 \in \mathit{dom}(\rho)$. Note that $\ell_{k-1} = \mathit{repl}(o_0[\tilde{\mathsf{a}}_0])$ also implies that $T \vdash \mathit{repl}(o_0[\tilde{\mathsf{a}}_0])$ and so $\mathit{repl}(o_0[\tilde{\mathsf{a}}_0]) \in \mathit{dom}(\rho)$. By definition of a matching mapping, we deduce that $\rho(ev_0) = \rho(\mathit{repl}(o_0[\tilde{\mathsf{a}}_0]))$. By our inductive hypothesis, we know that $T_{k-1}, k-1 \vdash^2 (\mathit{repl}(o_0[\tilde{\mathsf{a}}_0]), \mathit{repl}(o_1[\tilde{\mathsf{a}}_1])$ implies that $\rho(\mathit{repl}(o_0[\tilde{\mathsf{a}}_0])) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$ and so we conclude that $\rho(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$.

- *Other rules:* By Lemma 1, we directly obtain that there exists a biconfiguration $\mathcal{C}'_k$ such that $T_{k-1} \xrightarrow{(\ell_k,\ell'_k)}_{wi^2} \mathcal{C}'_k$ and $\mathit{proj}_0(\mathcal{C}'_k) = \mathcal{C}_k$. Let us denote $T_k = T_{k-1} \xrightarrow{(\ell_k,\ell'_k)}_{wi^2} \mathcal{C}'_k$. Hence $\mathit{proj}_0(T_k) = T_{|k}$.

  For all these rules, ℓ_k and ℓ'_k are not replication events so for all $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T_k \vdash^2 (ev_0, ev_1)$ implies that $T_{k-1} \vdash^2 (ev_0, ev_1)$. Hence we can directly conclude by our inductive hypothesis. □

Theorem 3 (May-testing preorder)**.** *If for all $T \in \mathsf{wtrace}(\mathit{proj}_0(\mathcal{C}_I))$, there exists a session matching ρ from P to Q such that:*

- $\{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev\} \subseteq \mathit{dom}(\rho)$
- *for all $C \in \mathbb{C}^{\mathrm{bad}}_{sat}$, $T, \rho \vdash_{cs} \mathsf{falsify}_0(C)$*

then $\pi_{\sqsubseteq_m}(\mathcal{C}_I)$ *holds.*

Proof. Let $C_0 = \mathit{projo}(\mathcal{C}_I)$. Let $T \in \mathsf{wtrace}(\mathcal{C}_0)$ with $T = \mathcal{C}_0 \xrightarrow{\ell_1}_i \dots \xrightarrow{\ell_n}_i \mathcal{C}_n$. We need to show that there exists a convergent bitrace $T' \in \mathsf{wtrace}^2(\mathcal{C})$ such that $\mathit{projo}(T') = T$.

By hypothesis, we know that there exists a matching mapping ρ from P to Q such that:

- for all $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T \vdash ev$ implies $ev \in \mathit{dom}(\rho)$
- for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $\rho \vdash_{cs} \mathsf{falsify}_0(C)$

Let us denote T_0 the empty bitrace from $\mathcal{C}_I$, i.e. $\mathcal{C}_I$ does not move. Since no events ev, ev' can be such that $T_0 \vdash^2 (ev, ev')$, we deduce that $\mathcal{P}_0(\rho, T_0)$ holds. Thus, by taking $k_0 = 0$, we have that $T_0 \in \mathsf{wrtrace}^2(\mathcal{C}_I)$, $\mathit{projo}(T_0) = T_{|0}$ and $\mathcal{P}_0(\rho, T_0)$. We can therefore apply Lemma 12 which allows us to deduce that there exists $T_n \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $\mathit{projo}(T_n) = T$, $T_{n|0} = T_0$ and $\mathcal{P}_0(\rho, T_n)$. By applying Lemma 5, we conclude that T_n converges and so the result holds. □

Theorem 4 (Observational preorder)**.** *If there exists a session matching ρ from P to Q such that:*

- $\mathit{dom}(\rho) = \{ev \in \mathsf{Ev}_{!o} \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(\mathit{projo}(\mathcal{C}_I))\}$
- *for all* $T \in \mathsf{wtrace}(\mathit{projo}(\mathcal{C}_I))$*, if* $\rho' = \rho \cup [\mathit{repl}(\bar{o}) \mapsto \rho(ev) \mid T \vdash \mathit{repl}_i(\bar{o}, M) = ev]$ *then for all* $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$*,* $T, \rho' \vdash_{cs} \mathsf{falsify}_0(C)$

then $\pi_{\sqsubseteq_o}(\mathcal{C}_I)$ *holds.*

Proof. Let us define the predicate on biconfigurations π such that $\pi(\mathcal{C})$ holds when there exists $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $|T| = n$, $T[n] = \mathcal{C}$ and for all $ev_0, ev_1 \in \mathsf{Ev}_{!o} \cup \mathsf{Ev}_{!i}$, if $T \vdash^2 (ev_0, ev_1)$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ then $ev_0 \in \mathit{dom}(\rho)$ and $\rho(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$.

To show that $\pi_{\sqsubseteq_o}(\mathcal{C}_I)$, we will show that $\pi(\mathcal{C}_I)$ holds and for all biconfigurations $\mathcal{C}$ such that $\pi(\mathcal{C})$ holds, we have

1. $\mathcal{C}$ converges
2. if $\mathit{projo}(\mathcal{C}) \xrightarrow{\ell_0}_{wi} \mathcal{C}'_0$ then $\mathcal{C} \xrightarrow{(\ell_0,\ell_1)}_{wi^2} \mathcal{C}'$, $\mathit{projo}(\mathcal{C}') = \mathcal{C}'_0$ and $\pi(\mathcal{C}')$ holds for some $\mathcal{C}', \ell_1$.

First, consider the empty trace $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$, i.e. the trace that does not move from $\mathcal{C}_I$. Hence $|T| = 0$ and $T[0] = \mathcal{C}_I$. Since no events ev, ev' can be such that $T' \vdash^2 (ev, ev')$, we deduce that $\pi(\mathcal{C}_I)$ holds.

Let now prove that for all biconfigurations $\mathcal{C}$, if $\pi(\mathcal{C})$ holds then $\mathcal{C}$ converges. Since $\pi(\mathcal{C})$ holds, there exists $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $|T| = n$, $T[n] = \mathcal{C}$ and for all $ev_0, ev_1 \in \mathsf{Ev}_{!o} \cup \mathsf{Ev}_{!i}$, if $T \vdash^2 (ev_0, ev_1)$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ then $ev_0 \in \mathit{dom}(\rho)$ and $\rho(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$.

Let us define $\rho' = \rho \cup [\mathit{repl}(\bar{o}) \mapsto \rho(ev) \mid \mathit{projo}(T) \vdash \mathit{repl}_i(\bar{o}, M) = ev]$. By hypothesis of the theorem, we know that for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $\rho' \vdash_{cs} \mathsf{falsify}_0(C)$. Moreover, by definition of $T \in \mathsf{wrtrace}^2(\mathcal{C}_I)$, for all $ev_0, ev_1 \in \mathsf{Ev}_!$ such that $T, i \vdash^2 (ev_0, ev_1)$, either $ev_0, ev_1 \in \mathsf{Ev}_{!o}$ or there exists $ev'_0, ev'_1 \in \mathsf{Ev}_{!i}$ such that $\mathsf{orepl}(ev_0) = \mathsf{orepl}(ev'_0)$, $\mathsf{orepl}(ev_1) = \mathsf{orepl}(ev'_1)$ and $T, i+1 \vdash^2 (ev'_0, ev'_1)$. Therefore, for all $ev_0, ev_1 \in \mathsf{Ev}_!$, if $T \vdash^2 (ev_0, ev_1)$ then $ev_0 \in \mathit{dom}(\rho')$. In the former case, we directly have that $\rho(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$ where $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ and so $\rho'(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$. In the latter case, we know that $\rho(ev'_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$

where $\mathsf{orepl}(ev'_1) = o_1[\tilde{\mathsf{a}}_1]$. By construction of ρ', $\rho(ev'_0) = \rho(ev_0)$. Therefore, $\rho'(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$ which allows to conclude that T converges by applying Lemma 5. As $T[n] = \mathcal{C}$, $\mathcal{C}$ also converges.

We now prove the final property. Assume that $proj_0(\mathcal{C}) \xrightarrow{\ell_0}_{wi} \mathcal{C}'_0$. Let us denote $T_e = proj_0(T) \xrightarrow{\ell_0}_{wi} \mathcal{C}'_0$. We have that $T_e \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$. Thus, let us denote by $\rho' = \rho \cup [repl(\overline{o}) \mapsto \rho(ev) \mid T_e \vdash repl_i(\overline{o}, M) = ev]$. We deduce by hypothesis that for all $C = (H \to bad) \in \mathbb{C}_{sat}$, $\rho' \vdash_{cs} \mathsf{falsify}_0(C)$. Moreover, by definition of $T_e \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$ and by construction of ρ', we deduce that for all $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T' \vdash ev$ implies $ev \in dom(\rho')$.

Similarly to above, we can show that for all $ev_0, ev_1 \in \mathsf{Ev}_!$ such that $T \vdash^2 (ev_0, ev_1)$, we have that $\rho'(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$ where $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ and so $\rho'(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$, which allows us to deduce that $\mathcal{P}_0(\rho', T)$.

As $|T| = n$ and $|T_e| = n+1$ or $|T_e| = n+2$, we apply Lemma 12 with $k_0 = n$ and $T_{k_0} = T$, which allows us to obtain that there exists $T' \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $proj_0(T') = T_e$, $T'_{|n} = T$ and $\mathcal{P}_0(\rho', T')$. Since $T'_{|n} = T$ and $proj_0(T') = T_e$, we deduce that $\mathcal{C} \xrightarrow{(\ell_0,\ell_1)}_{wi^2} T'[|T_e|]$ and $proj_0(T'[|T_e|]) = \mathcal{C}'_0$. Finally, as $\mathcal{P}_0(\rho', T')$ and $dom(\rho) = dom(\rho')_{|\mathsf{Ev}_{!i} \cup \mathsf{Ev}_{!o}}$, we conclude that $\pi(T'[|T_e|])$ holds. □

Theorem 5 (Observational equivalence). *If there exists two matching mapping ρ_0 and ρ_1 from P to Q and Q to P respectively such that for all $i \in \{0,1\}$,*

- *$dom(\rho_i) = \{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))\}$*
- *for all $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$, for all $C = (H \to \mathrm{bad}) \in \mathbb{C}_{sat}$, $T, \rho_i \vdash_{cs} \mathsf{falsify}_i(C)$*
- *for all $ev_0 \in dom(\rho_0)$, for all $ev_1 \in dom(\rho_1)$, if $\mathsf{orepl}(ev_0) = o_0[\tilde{\mathsf{a}}_0]$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ then $o_0[\tilde{\mathsf{a}}_{0|\lambda}] = \rho_1(ev_1)$ iff $o_1[\tilde{\mathsf{a}}_{1|\lambda}] = \rho_0(ev_0)$*

then $\pi_{\approx_o}(\mathcal{C}_I)$ holds.

Proof. Let us define the predicate on biconfigurations π such that $\pi(\mathcal{C})$ holds when there exists $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $|T| = n$, $T[n] = \mathcal{C}$, $\mathcal{P}_0(\rho_0, T)$ and $\mathcal{P}_1(\rho_1, T)$.

To show that $\pi_{\approx_o}(\mathcal{C}_I)$, we will show that $\pi(\mathcal{C}_I)$ holds and for all biconfigurations $\mathcal{C}$ such that $\pi(\mathcal{C})$ holds, we have that

1. $\mathcal{C}$ converges
2. for all $i \in \{0,1\}$, if $proj_i(\mathcal{C}) \xrightarrow{\ell_i}_{wi} \mathcal{C}'_i$ then $\mathcal{C} \xrightarrow{(\ell_0,\ell_1)}_{wi^2} \mathcal{C}'$, $proj_i(\mathcal{C}') = \mathcal{C}'_i$ and $\pi(\mathcal{C}')$ holds for some $\mathcal{C}', \ell_{1-i}$.

First, consider the empty trace $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$, i.e. the trace that does not move from $\mathcal{C}_I$. Hence $|T| = 0$ and $T[0] = \mathcal{C}_I$. Since no events ev, ev' can be such that $T' \vdash^2 (ev, ev')$, we deduce that $\pi(\mathcal{C}_I)$ holds.

Let now prove that for all biconfigurations $\mathcal{C}$, if $\pi(\mathcal{C})$ holds then $\mathcal{C}$ converges. Since $\pi(\mathcal{C})$ holds, there exists $T \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $|T| = n$, $T[n] = \mathcal{C}$, $\mathcal{P}_0(\rho_0, T)$ and $\mathcal{P}_1(\rho_1, T)$. Moreover by hypothesis, for all $C = (H \to \mathrm{bad}) \in \mathbb{C}_{sat}$, $\rho_0 \vdash_{cs} \mathsf{falsify}_0(C)$. By Lemma 5 we deduce that T converges and so $T[n]$ converges too.

Let $i \in \{0,1\}$ such that $proj_i(\mathcal{C}) \xrightarrow{\ell_i}_{wi} \mathcal{C}'_i$. Let $T_e = proj_i(T) \xrightarrow{\ell_i}_{wi} \mathcal{C}'_i$. Hence, $T_e \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$ with $|T_e| = n_e$. By hypothesis of the theorem, we know that for all $ev \in$

$\mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$, $T_e \vdash ev$ implies $ev \in dom(\rho_i)$. Moreover, we also know that for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $\rho \vdash_{cs} \mathsf{falsify}_i(C)$. Therefore, by taking $k_0 = n$, $T_{k_0} = T$, we have $proj_i(T_{k_0}) = T_{e|k_0}$ and $\mathcal{P}_i(\rho_i, T)$. This allows us to apply Lemma 12 which implies that there exists $T' \in \mathsf{wtrace}^2(\mathcal{C}_I)$ such that $proj_i(T') = T_e$, $T'_{|n} = T$ and $\mathcal{P}_i(\rho_i, T')$.

It remains to show that $\mathcal{P}_{1-i}(\rho_{1-i}, T')$ holds. Let $ev_0, ev_1 \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ such that $T' \vdash^2 (ev_0, ev_1)$. Let us assume that $\mathsf{orepl}(ev_0) = o_0[\tilde{\mathsf{a}}_0]$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$. We know that $ev_0 \in dom(\rho_0)$ and $ev_1 \in dom(\rho_1)$. We know that $\mathcal{P}_i(\rho_i, T')$ holds, hence $\rho_i(ev_i) = o_{1-i}[\tilde{\mathsf{a}}_{1-i|\lambda}]$. Therefore, by the last hypothesis of the theorem, we deduce that $o_i[\tilde{\mathsf{a}}_{i|\lambda}] = \rho_{1-i}(ev_{1-i})$. We can conclude that $\mathcal{P}_{1-i}(\rho_{1-i}, T')$ holds. □

E Proofs of practical verification (Theorems 6 to 8)

For this proof, we will rely on the fact that the set of terms is countable. We denote by θ the bijection from terms to $\mathbb{N}$.

Theorem 6 (May-testing preorder). *If for all $C \in \mathbb{C}_{sat}^{\text{bad}}$, we can associate a skolemisation substitution σ_C of $\mathsf{falsify}_0(C)$ such that for all $C_0, C_1 \in \mathbb{C}$ (C_0, σ_{C_0} and C_1, σ_{C_1} are renamed such that they have distinct variables) with $\mathsf{falsify}_0(C_i) = (\Omega_i, \phi_i)$ and $H'_i = proj_0(H_i)$, for all $ev_i \in dom(\Omega_i)$ for $i = 0, 1$, we have:*

1. *if $(ev'_0, ev'_1) = \mathsf{same}(ev_0, ev_1)$, $\alpha = mgu(ev'_0, ev'_1)$ then:*

$$(H'_0 \wedge H'_1 \wedge \Omega(ev_0)\sigma_{C_0} \neq \Omega(ev_1)\sigma_{C_1} \to \text{bad})\alpha{\downarrow} = \bot$$

2. *if $\alpha = mgu(\Omega(ev_0)\sigma_{C_0}, \Omega(ev_1)\sigma_{C_1})$ then:*

$$(H'_0 \wedge H'_1 \wedge \mathsf{orepl}(ev_0)_{|\lambda} \neq \mathsf{orepl}(ev_1)_{|\lambda} \to \text{bad})\alpha{\downarrow} = \bot$$

then $\pi_{\sqsubseteq_m}(\mathcal{C}_I)$ holds.

Proof. To prove that $\pi_{\approx_m}(\mathcal{C}_I)$ holds, we will rely on Theorem 3. Let $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$. We need to build a session matching ρ from P to Q such that:

- $dom(\rho) = \{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev\}$
- for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $\rho \vdash_{cs} \mathsf{falsify}_0(C)$

Consider $\{C_i = (H_i \to \text{bad})\}_{i=1}^n = \mathbb{C}_{sat}^{\text{bad}}$ and lets consider $\sigma_1, \ldots, \sigma_n$ their Skolemization.

We start by building ρ_0 as follows: If there exists a substitution γ and $i \in \{1, \ldots, n\}$ such that $T \vdash H_i\gamma$ then for all $ev \in dom(\Omega_i)$, if $\Omega_i(ev)\sigma_i = o[M_1, \ldots, M_k]$ then $\rho_0(ev\gamma) = o[\theta(M_1\gamma), \ldots, \theta(M_k\gamma)]$.

Note that for ρ_0 to be a mapping, our definition should ensure that no two different occurrence pattern names are given for the same argument. In other word, we need to show that if there exist two substitutions γ_1, γ_2 and $i_1, i_2 \in \{1, \ldots, n\}$ and $ev_1 \in \Omega_{i_1}$ and $ev_2 \in \Omega_{i_2}$ with $y_1 = \Omega_{i_1}(ev_1)$ and $y_2 = \Omega_{i_2}(ev_2)$ such that

- $T \vdash H_{i_1}\gamma_1$, $T \vdash H_{i_2}\gamma_2$
- $y_1\sigma_{i_1} = o_1[M_1, \ldots, M_{k_1}]$

- $y_2\sigma_{i_2} = o_2[N_1, \ldots, N_{k_2}]$
- $ev_1\gamma_1 = ev_2\gamma_2$

then $o_1[\theta(M_1\gamma_1), \ldots, \theta(M_{k_1}\gamma_1)] = o_2[\theta(N_1\gamma_2), \ldots, \theta(N_{k_2}\gamma_2)]$.

Let's assume the four hypotheses and assume by contradiction that $o_1[\theta(M_1\gamma_1), \ldots, \theta(M_{k_1}\gamma_1)] \neq o_2[\theta(N_1\gamma_2), \ldots, \theta(N_{k_2}\gamma_2)]$. Since θ is a bijection, we have $o_1[M_1\gamma_1, \ldots, M_{k_1}\gamma_1] \neq o_2[N_1\gamma_2, \ldots, N_{k_2}\gamma_2]$.

Since $ev_1\gamma_1 = ev_2\gamma_2$ and ev_1, ev_2 have distinct variables, we deduce that ev_1 and ev_2 are unifiable and so if $\alpha = mgu(ev_1, ev_2)$ then $\gamma_1\gamma_2 = \gamma_2\gamma_1 = \alpha\gamma$ and

- $o_1[M_1\gamma_1, \ldots, M_{k_1}\gamma_1] = o_1[M_1, \ldots, M_{k_1}]\alpha\gamma$ and $o_2[N_1\gamma_2, \ldots, N_{k_2}\gamma_2] = o_2[N_1, \ldots, N_{k_2}]\alpha\gamma$
- $ev_1\gamma_1 = ev_1\alpha\gamma$ and $ev_2\gamma_2 = ev_2\alpha\gamma$
- $H_{i_1}\gamma_1 = H_{i_1}\alpha\gamma$ and $H_{i_2}\gamma_2 = H_{i_2}\alpha\gamma$

Thus, $\gamma \models o_1[M_1, \ldots, M_{k_1}]\alpha \neq o_2[N_1, \ldots, N_{k_2}]\alpha$ and $T \vdash H_{i_1}\alpha\gamma \wedge H_{i_2}\alpha\gamma$. This is in contradiction with the first hypothesis of the theorem. Hence, $o_1[\theta(M_1\gamma_1), \ldots, \theta(M_{k_1}\gamma_1)] = o_2[\theta(N_1\gamma_2), \ldots, \theta(N_{k_2}\gamma_2)]$.

Let us show that ρ_0 is matching mapping from P to Q. We need to verify the four bullet points of Definition 16.

- Direct by construction.
- The second bullet point is directly derived from Item 2 of Definition 21 and by construction of ρ_0
- For the third bullet point, we can apply the a similar proof to the one showing that no two different occurrence patterns are given for the same argument .
- Consider $ev_1, ev_2 \in dom(\rho_0)$ such that $\rho(ev_1) = \rho(ev_2)$. By contradiction, assume that $ev_1 \neq ev_2$. By definition, $ev_1, ev_2 \in dom(\rho_0)$ implies there exists γ_1, γ_2 and $i_1, i_2 \in \{1, \ldots, n\}$ such that for all $j \in \{1, 2\}$, $T \vdash H_{i_j}\gamma_j$ and there exists $(ev'_j, \phi_j, y_j) \in \mathsf{M}_j$ such that $y_j\sigma_j = o_j[M^j_1, \ldots, M^j_{k_j}]$, $ev_j = ev'_j\gamma_j$ and $\rho_0(ev_j) = o_j[\theta(M^j_1\gamma_j), \ldots, \theta(M^j_{k_j}\gamma_j)]$.

  Since $\rho_0(ev_1) = \rho_0(ev_2)$, we deduce that $o_1 = o_2$, $k_1 = k_2$ and $M^1_\ell\gamma_1 = M^2_\ell\gamma_2$ for all $j \in \{1, \ldots, k_1\}$. As the variables of $o_j[M^j_1, \ldots, M^j_{k_j}]$, $j = 1, 2$ are distincts, we deduce that $o_1[M^1_1, \ldots, M^1_{k_1}]$ and $o_2[M^2_1, \ldots, M^2_{k_2}]$ are unifiable. Let us denote by α their most general unifier. There exists γ such that $\gamma_1\gamma_2 = \gamma_2\gamma_1 = \alpha\gamma$ and:

  - $o_1[M^1_1\gamma_1, \ldots, M^1_{k_1}\gamma_1] = o_1[M^1_1, \ldots, M^1_{k_1}]\alpha\gamma = y_1\sigma_1\alpha\gamma$ and $o_2[M^2_1\gamma_2, \ldots, M^2_{k_2}\gamma_2] = o_2[M^2_1, \ldots, M^2_{k_2}]\alpha\gamma = y_2\sigma_2\alpha\gamma$
  - $ev_1 = ev'_1\gamma_1 = ev'_1\gamma_1\gamma_2 = ev'_1\alpha\gamma$ and $ev_2 = ev'_2\gamma_2 = ev'_2\gamma_2\gamma_1 = ev'_2\alpha\gamma$ and $ev_1 \neq ev_2$
  - $H_{i_1}\gamma_1 = H_{i_1}\alpha\gamma$ and $H_{i_2}\gamma_2 = H_{i_2}\alpha\gamma$

  Thus, $\gamma \models ev'_1\alpha \neq ev'_2\alpha$ and $T \vdash H_{i_1}\alpha\gamma \wedge H_{i_2}\alpha\gamma$. This is in contradiction with the second hypothesis of the theorem. Hence, $ev_1 = ev_2$.

This allows us to conclude that ρ_0 is a matching mapping from P to Q.

It remains to show that for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $T, \rho_0 \vdash_{cs} \mathsf{falsify}_0(C)$. Let $\mathsf{falsify}_0(C) = (\Omega, \phi')$. Let $H' \wedge \phi = proj_0(H)$. Let γ be a substitution such that $T \vdash H'\gamma \wedge \phi\gamma$ and for all $ev \in dom(\Omega)$ with $y = \Omega(ev)$, $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ implies $ev\gamma \in dom(\rho_0)$ and $y\gamma = \rho(ev\gamma)$. By construction, we know that there exists a skolemization σ such that $y\sigma = o[M_1, \dots, M_k]$ implies $y\gamma = o[M_1\gamma, \dots, M_k\gamma]$. As $y\sigma\gamma = y\gamma$ and by construction of ϕ', we have that $\phi'\sigma\gamma = \phi'\gamma$. By definition of a skolemization, we know that $\phi \models \phi'\sigma$ and so we conclude that $\models \phi'\gamma$. □

Theorem 7 (Observational preorder)**.** *If for all $C \in \mathbb{C}_{sat}^{\text{bad}}$, we can associate a skolemisation substitution σ_C of $\mathsf{falsify}_0(C)$ such that for all $C_i = (H_i \wedge \phi'_i \to \text{bad}) \in \mathbb{C}_{sat}^{\text{bad}}$ (taking C_i and σ_{C_i} renamed) with $\mathsf{falsify}_0(C_i) = (\Omega_i, \phi_i)$, for all $ev_i \in dom(\Omega_i)$ with $\Phi_i = \phi'_{i|vars(ev_i)}$ for $i = 0, 1$, we have:*

1. *if $(ev'_0, ev'_1) = \mathsf{same}(ev_0, ev_1)$, $\alpha = mgu(ev'_0, ev'_1)$ then:*

$$(\Phi_0 \wedge \Phi_1 \wedge \Omega(ev_0)\sigma_{C_0} \neq \Omega(ev_1)\sigma_{C_1} \to \text{bad})\alpha{\downarrow} = \bot$$

2. *if $\alpha = mgu(\Omega(ev_0)\sigma_{C_0}, \Omega(ev_1)\sigma_{C_1})$ then:*

$$(\Phi_0 \wedge \Phi_1 \wedge \mathsf{orepl}(ev_0)_{|\lambda} \neq \mathsf{orepl}(ev_1)_{|\lambda} \to \text{bad})\alpha{\downarrow} = \bot$$

then $\pi_{\sqsubseteq_o}(\mathcal{C})$ holds.

Proof. To prove that $\pi_{\sqsubseteq_o}(\mathcal{C}_I)$ holds, we will rely on Theorem 4. Let $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$. We need to build a matching mapping ρ from P to Q such that:

- $\{ev \in \mathsf{Ev}_{!o} \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))\} \subseteq dom(\rho)$
- for all $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, if $\rho' = \rho \cup [repl(\overline{o}) \mapsto \rho(ev) \mid T \vdash repl_i(\overline{o}, M) = ev]$ then for all $C = (H \to \text{bad}) \in \mathbb{C}_{sat}$, $T, \rho' \vdash_{cs} \mathsf{falsify}_0(C)$

We start by building ρ_0 as follows: If there exist $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, a substitution γ and $C \in \mathbb{C}$ with $\mathsf{falsify}_0(C) = (\Omega, \phi)$, $H = proj_0(C)$ such that $T \vdash H\gamma$ then for all $ev \in \Omega$ with $y = \Omega(ev)$, $ev\gamma \in dom(\rho_0)$ and $\rho_0(ev\gamma) = y\sigma_C\gamma$.

Note that for ρ_0 to be a mapping, our definition should ensure that no two different occurrence pattern names are given for the same argument. In other word, we need to show that if there exist two traces $T_1, T_2 \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, two substitutions γ_1, γ_2, two clauses C_1, C_2 with $\mathsf{falsify}_0(C_1) = (\Omega_1, \phi_1)$ with $H_1 = proj_0(C_1)$ and $\mathsf{falsify}_0(C_2) = (\Omega_2, \phi_2)$ with $H_2 = proj_0(C_2)$, and $ev_1 \in dom(\Omega_1)$ with $y_1 = \Omega_1(ev_1)$ and $\phi_{ev_1} = \phi(C_1)_{|vars(ev_1)}$, and $ev_2 \in dom(\Omega_2)$ with $y_2 = \Omega_2(ev_2)$ and $\phi_{ev_2} = \phi(C_2)_{|vars(ev_2)}$ such that

- $T_1 \vdash H_1\gamma_1$, $T_2 \vdash H_2\gamma_2$
- $ev_1\gamma_1 = ev_2\gamma_2$

then $y_1\sigma_{C_1}\gamma_1 = y_2\sigma_{C_2}\gamma_2$.

Let us assume the two hypotheses and assume by contradiction that $y_1\sigma_{C_1}\gamma_1 \neq y_2\sigma_{C_2}\gamma_2$. Since $ev_1\gamma_1 = ev_2\gamma_2$ and ev_1, ev_2 have distinct variables, we deduce that ev_1, ev_2 are unifiable and so if $\alpha = mgu(ev_1, ev_2)$ then $\gamma_1\gamma_2 = \alpha\gamma$ for some γ. By construction, for $i \in \{1, 2\}$, if

$C_i = (H'_i \wedge \phi'_i)$ then $\phi_{ev_i} = \phi'_{i|fv(()ev_i)}$. Furthermore, $H_1 = proj_0(H'_i) \wedge \phi'_{i|\mathsf{X}_i}$ where $fv(ev_i) \subseteq \mathsf{X}_i$. Hence, $T_i \vdash H_i\gamma_i$ implies that $\models \phi_{ev_i}\gamma_i$. As $\phi_{ev_i}\gamma_i = \phi_{ev_i}\alpha\gamma$, we deduce that $\gamma \models \phi_{ev_i}\alpha$.

We also have for $i \in \{1,2\}$, $y_i\sigma_{C_i}\gamma_i = y_i\sigma_{C_i}\alpha\gamma$. Hence, $\gamma \models y_1\sigma_{C_1}\alpha \neq y_2\sigma_{C_2}\alpha$. This allows us to deduce that $\gamma \models \phi_{ev_1}\alpha \wedge \phi_{ev_2}\alpha \wedge y_1\sigma_{C_1}\alpha \neq y_2\sigma_{C_2}\alpha$ which is in contradiction with the first hypothesis of the theorem. We conclude that $y_1\sigma_{C_1}\gamma_1 = y_2\sigma_{C_2}\gamma_2$.

Let us show that ρ_0 is matching mapping from P to Q. We need to verify the three bullet points of Definition 16.

- Direct by construction.
- The second bullet point is directly derived from Item 2 of Definition 21 and by construction of ρ_0
- For the third bullet point, we can apply the a similar proof to the one showing that no two different occurrence patterns are given for the same argument .
- Consider $ev_1, ev_2 \in dom(\rho_0)$ such that $\rho(ev_1) = \rho(ev_2)$. By contradiction, assume that $ev_1 \neq ev_2$. By definition, $ev_1, ev_2 \in dom(\rho_0)$ implies there exist $T_1, T_2 \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, two substitutions γ_1, γ_2 and $C_1, C_2 \in \mathbb{C}$ with $\mathsf{falsify}_0(C_1) = (\Omega_1, \phi_1)$ with $H_1 = proj_{0(C_1)}$ and $\mathsf{falsify}_0(C_2) = (\Omega_2, \phi_2)$ with $H_2 = proj_{0(C_2)}$, and $ev'_1 \in dom(\Omega_1)$ with $y_1 = \Omega_1(ev'_1)$ and $\phi_{ev'_1} = \phi(C_1)_{|vars(ev'_1)}$, and $ev'_2 \in dom(\Omega_2)$ with $y_2 = \Omega_2(ev'_2)$ and $\phi_{ev'_2} = \phi(C_2)_{|vars(ev'_2)}$ such that for all $i \in \{1,2\}$, $T_i \vdash H_i\gamma_i$ and $\rho(ev_i) = y_i\sigma_{C_i}\gamma_i$.

  Since $\rho_0(ev_1) = \rho_0(ev_2)$ and the variables of σ_{C_1} and σ_{C_2} are distinct, we deduce that $y_1\sigma_{C_1}$ and $y_2\sigma_{C_2}$ are unifiable. Let $\alpha = mgu(y_1\sigma_{C_1}, y_2\sigma_{C_2})$. There exists γ such that $\gamma_1\gamma_2 = \gamma_2\gamma_1 = \alpha\gamma$ and:

  - $y_1\sigma_{C_1}\alpha\gamma = y_2\sigma_{C_2}\alpha\gamma$
  - $ev_1 = ev'_1\gamma_1 = ev'_1\alpha\gamma$ and $ev_2 = ev'_2\gamma_2 = ev'_2\alpha\gamma$
  - $H_1\gamma_1 = H_1\alpha\gamma$ and $H_2\gamma_2 = H_2\alpha\gamma$

  Once again by construction of H_1 and H_2, $T_1 \vdash H_1\gamma_1$ and $T_2 \vdash H_2\gamma_2$ implies that $\models \phi_{ev'_1}\gamma_1$ and $\models \phi_{ev'_2}\gamma_2$. Hence $\gamma \models \phi_{ev'_1}\alpha \wedge \phi_{ev'_2}\gamma \wedge ev'_1\alpha \neq ev'_2\alpha$. This is in contradiction with the second hypothesis of the theorem. Hence, $ev_1 = ev_2$.

This allows us to conclude that ρ_0 is a matching mapping from P to Q.

It remains to show that for all $T \in \mathsf{wtrace}(proj_0(\mathcal{C}_I))$, if $\rho' = \rho_0 \cup [repl(\overline{o}) \mapsto \rho_0(ev) \mid T \vdash repl_i(\overline{o}, M) = ev]$ then for all $C = (H \rightarrow \text{bad}) \in \mathbb{C}_{sat}$, $T, \rho' \vdash_{cs} \mathsf{falsify}_0(C)$. Notice that if $repl_i(\overline{o}, M) \in dom(\rho_0)$ and $repl(\overline{o}) \in dom(\rho_0)$ then by item 3 of Definition 21, we know that $\rho_0(repl_i(\overline{o}, M)) = \rho_0(repl(\overline{o}))$. Hence, ρ' is also a matching mapping from P to Q.

More specifically, if $\mathsf{falsify}_0(C) = (\Omega, \phi)$ with $H = proj_0(C)$ and γ is a substitution such that $T \vdash H\gamma$ and for all $ev \in dom(\Omega)$ with $y = \Omega(ev)$ and $\phi_{ev} = \phi(C)_{|vars(ev)}$, $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ implies $ev\gamma \in dom(\rho')$ and $y\gamma = \rho(ev\gamma)$ then by construction, we know that $ev\gamma \in dom(\rho_0)$ and $\rho(ev\gamma) = y\sigma_C\gamma$. As $y\sigma_C\gamma = y\gamma$ and by construction of ϕ, we have that $\phi\sigma_C\gamma = \phi\gamma$. By definition of a skolemization, we know that if $H = H' \wedge \phi'$ then $T \vdash H\gamma$ implies that $\models \phi'\gamma$ which implies that $\models \phi\sigma_C\gamma$ and so $\models \phi\gamma$. □

Theorem 8 (Observational equivalence)**.** *If there exists a bijection β_0 from the occurrence replication names of P to the ones of Q (we denote β_1 its inverse) such that for all $i \in \{0,1\}$, for all $o_i, o'_i \in dom(\rho_i)$,* *1.* $\beta_i(o_i) = o_{1-i}$ *implies* $(o_0, o_1) \in \mathrm{pm}(\mathcal{C})$ *and* $\mathsf{ar}^{\lambda}_{\mathcal{C}_I}(o_0) = \mathsf{ar}^{\lambda}_{\mathcal{C}_I}(o_1)$

2. $o_i \preceq_{\mathcal{C}_I} o'_i$ *if and only if* $\beta_i(o_i) \preceq_{\mathcal{C}_I} \beta_i(o'_i)$

and for all $C = (H \wedge \phi \rightarrow \mathrm{bad}) \in \mathbb{C}^{\mathrm{bad}}_{sat}$ *such that we have* $\mathsf{falsify}_i(C) = (\Omega, \phi')$*,*

3. if $\sigma = \{\Omega(ev) \mapsto \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}] \mid ev \in dom(\Omega) \wedge \mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]\}$ *then* $\phi_{|vars_i(C)} \models \phi'\sigma$*.*

then $\pi_{\approx_o}(\mathcal{C})$ *holds.*

Proof. To prove that $\pi_{\approx_o}(\mathcal{C}_I)$ holds, we will rely on Theorem 5. Therefore, we need to build two matching mapping ρ_0 and ρ_1 from P_0 to P_1 and P_1 to P_0 respectively such that for all $i \in \{0, 1\}$,

- $dom(\rho_i) = \{ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i} \mid T \vdash ev \wedge T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))\}$
- for all $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$, for all $C = (H \rightarrow \mathrm{bad}) \in \mathbb{C}_{sat}$, $T, \rho_i \vdash_{cs} \mathsf{falsify}_i(C)$
- for all $ev_0 \in dom(\rho_0)$, for all $ev_1 \in dom(\rho_1)$, if $\mathsf{orepl}(ev_0) = o_0[\tilde{\mathsf{a}}_0]$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ then $o_0[\tilde{\mathsf{a}}_{0|\lambda}] = \rho_1(ev_1)$ iff $o_1[\tilde{\mathsf{a}}_{1|\lambda}] = \rho_0(ev_0)$

Let us build ρ_0 and ρ_1 as follows: For all $i \in \{0, 1\}$, for all $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$, if $T \vdash ev$ with $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$ then $ev \in dom(\rho_i)$ and $\rho_i(ev) = \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}]$.

We show that for all $i \in \{0, 1\}$, ρ_i is a matching mapping from P_i to P_{1-i}. We need to verify the three bullet point of Definition 16.

- Direct by construction and by the first hypothesis of the theorem.
- Direct once again by construction and by the second hypothesis of the theorem.
- Consider $ev_1, ev_2 \in dom(\rho_i)$ such that $\rho_i(ev_1) = \rho_i(ev_2)$ with $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$ and $\mathsf{orepl}(ev_2) = o_2[\tilde{\mathsf{a}}_2]$. Since $\rho_i(ev_1) = \rho_i(ev_2)$, we deduce that $\beta_i(o_1)[\tilde{\mathsf{a}}_{1|\lambda}] = \beta_i(o_2)[\tilde{\mathsf{a}}_{2|\lambda}]$. This implies that $\beta_i(o_1) = \beta_i(o_2)$ and $\tilde{\mathsf{a}}_{1|\lambda} = \tilde{\mathsf{a}}_{2|\lambda}$. As β_i is a bijection, we deduce that $o_1 = o_2$ which allows us to conclude.

Let us now show the second bullet point of Theorem 5. Let $T \in \mathsf{wtrace}(proj_i(\mathcal{C}_I))$. Let $C \in \mathbb{C}_{sat}$ with $\mathsf{falsify}_i(\mathcal{C}_I) = (\Omega, \phi')$ with $H \wedge \phi = proj_i(C)$. Let γ be a substitution such that $T \vdash H\gamma \wedge \phi\gamma$ and for all $(ev, \phi_{ev}, y) \in \mathsf{M}$, $ev \in \mathsf{Ev}_! \cup \mathsf{Ev}_{!i}$ implies $ev\gamma \in dom(\rho_i)$ and $y\gamma = \rho_i(ev\gamma)$. Note that if $\mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]$ then by definition $\rho_i(ev\gamma) = \beta_i(o)[\tilde{\mathsf{a}}\gamma_{|\lambda}] = \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}]\gamma$.

Thus, if we define $\sigma = \{y \mapsto \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}] \mid (ev, \phi_{ev}, y) \in \mathsf{M} \wedge \mathsf{orepl}(ev) = o[\tilde{\mathsf{a}}]\}$, we obtain that $y\gamma = \beta_i(o)[\tilde{\mathsf{a}}_{|\lambda}]\gamma = y\sigma\gamma$. Thus, by construction $\phi'\sigma\gamma = \phi'\gamma$. However by the third hypothesis of the theorem, we know that $\phi \models \phi'\sigma$. Since $T \vdash H\gamma \wedge \phi\gamma$, we deduce that $\models \phi\gamma$ and so $\models \phi'\sigma\gamma$ and so $\models \phi'\gamma$. This conclude the proof that $T, \rho_i \vdash_s \mathsf{falsify}_i(\mathcal{C}_I)$.

Let us show the last bullet point of Theorem 5. Consider $ev_0 \in dom(\rho_0)$ and $ev_1 \in dom(\rho_1)$ such that $\mathsf{orepl}(ev_0) = o_0[\tilde{\mathsf{a}}_0]$ and $\mathsf{orepl}(ev_1) = o_1[\tilde{\mathsf{a}}_1]$. By definition of ρ_0 and ρ_1, we have that $o_0[\tilde{\mathsf{a}}_{0|\lambda}] = \rho_1(ev_1)$ iff $o_0[\tilde{\mathsf{a}}_{0|\lambda}] = \beta_1(o_1)[\tilde{\mathsf{a}}_{1|\lambda}]$ iff ($o_0 = \beta_1(o_1)$ and $\tilde{\mathsf{a}}_{0|\lambda} = \tilde{\mathsf{a}}_{1|\lambda}$) iff ($\beta_0(o_0) = o_1$ and $\tilde{\mathsf{a}}_{0|\lambda} = \tilde{\mathsf{a}}_{1|\lambda}$) iff $\beta_0(o_0)[\tilde{\mathsf{a}}_{0|\lambda}] = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$ iff $\rho_0(ev_0) = o_1[\tilde{\mathsf{a}}_{1|\lambda}]$. □