DEEPSEC: Deciding Equivalence Properties in Security Protocols
Theory and Practice

Vincent Cheval[1], Steve Kremer[1], and Itsaka Rakotonirina[1]

[1]Inria Nancy, LORIA, France

Abstract

We study the automated verification of behavioural equivalences in the applied pi calculus, an essential problem in formal, symbolic analysis of cryptographic protocols. We establish new complexity results for static equivalence, trace equivalence and labelled bisimilarity and propose a new decision procedure for these equivalences. Our procedure is the first tool to decide trace equivalence and labelled bisimilarity exactly for a family of equational theories, namely those that can be represented by a subterm convergent destructor rewrite system. Finally, we implement the procedure in a new tool, called DEEPSEC and demonstrate the applicability of the tool on several case studies.

1 Introduction

The use of automated, formal methods has become indispensable for analysing complex security protocols, such as those for authentication, key exchange and secure channel establishment. Nowadays there exist mature, fully automated such analysers; among others AVISPA [ABB+05], ProVerif [BSC16], Scyther [Cre08], Tamarin [SMCB13] or Maude-NPA [SEMM14]. These tools operate in so-called *symbolic* models, rooted in the seminal work by Dolev and Yao [DY81]: the attacker has full control of the communication network, unbounded computational power, but cryptography is idealised. This model is well suited for finding attacks in the protocol logic, and tools have indeed been extremely effective in discovering this kind of flaw or proving their absence.

While most works investigate *reachability* properties, a recent trend consists in adapting the tools—and the underlying theory—for the more complex *indistinguishability* properties. Such properties are generally modelled as a behavioural equivalence (bisimulation or trace equivalence) in a dedicated process calculus such as the Spi [AG99] or applied pi calculus [AF04]. A typical example is real-or-random secrecy: after interacting with a protocol, an adversary is unable to distinguish the *real* secret used in the protocol from a *random* value. Privacy-type properties can also be expressed as such: anonymity may be modeled as the adversary's inability to distinguish two instances of a protocol executed by different agents; vote privacy [DKR09] has been expressed as indistinguishability of the situations where the votes of two agents have been swapped or not; unlinkability [ACRR10] is seen as indistinguishability of two sessions, either both executed by the same agent A, or by two different agents A and B.

Related work

The problem of analysing security protocols is undecidable in general but several decidable subclasses have been identified. While many complexity results are known for trace properties [DLM04, RT03], the case of behavioural equivalences remains mostly open. When the attacker is an eavesdropper and cannot interact with the protocol, the indistinguishability problem—*static equivalence*—has been shown PTIME for large classes of cryptographic primitives [AC06, CDK12, CBC10]. For active attackers, bounding the number of protocol sessions is often sufficient to obtain decidability [RT03] and is of practical interest: most real-life attacks indeed only require a small number of sessions. In this context Baudet [Bau05], and later Chevalier and Rusinowtich [CR10], showed that real-or-random secrecy was coNP for cryptographic primitives that can be modelled as subterm convergent rewrite systems, by checking whether two constraint systems admit the same set of solutions. These procedures do however not allow for else branches, nor do they verify trace equivalence in full generality. In [CCD13], Cheval et al. have used Baudet's procedure as a black box to verify trace equivalence of *determinate* processes. This class of processes is however insufficient for most anonymity properties. Finally, decidability results for an unbounded number of sessions were proposed in [CCD15b, CCD15a], but with severe restrictions on processes and equational theories.

Tool support also exists for verifying equivalence properties. We start discussing tools that are limited to a bounded number of sessions. The SPEC tool [TD10, TNH16] verifies a sound symbolic bisimulation, but is restricted to particular cryptographic primitives (pairing, encryption, signatures and hash functions) and does not allow for else branches. The APTE tool [Che14] covers the same primitives but allows else branches and decides trace equivalence exactly. On the contrary, the AKISS tool [CCCK16] allows for user-defined cryptographic primitives. Partial correctness of AKISS is shown for primitives modelled by an arbitrary convergent rewrite system that has the finite variant property [CD05]. Termination is additionally shown for subterm convergent rewrite systems. However, AKISS does only decide trace equivalence for a class of determinate processes; for other processes trace equivalence can be both over- and under-approximated which proved to be sufficient on many examples. The recent SAT-EQUIV tool [CDD17] uses a different approach: it relies on Graph Planning and SAT solving to verify trace equivalence, rather than a dedicated procedure. The tool is extremely efficient and several orders of magnitude faster than other tools. It does however not guarantee termination and is currently restricted to pairing and symmetric encryption and only considers a class of *simple processes* (a subclass of determinate processes) that satisfy a type-compliance condition. These restrictions severely limit its scope.

Several other tools support verification of equivalence properties, even for an unbounded number of sessions. This is the case of ProVerif [BAF05], Tamarin [BDS15] and Maude NPA [SEMM14] which all allow for user-defined cryptographic primitives. However, given that the underlying problem is undecidable, these tools may not terminate. Moreover, they only approximate trace equivalence by verifying a more fine-grained equivalence, called *diff-equivalence*. This equivalence is too fine-grained on many examples. While some recent improvements on ProVerif [CB13, BS16] helps covering more protocols, general verification of trace equivalence is still out of scope. For instance, the verification by Arapinis et al. [AMR^{+}12] of unlinkability in the 3G mobile phone protocols required some "tricks" and approximations of the protocol to avoid false attacks. In [CGLM17], Cortier et al. develop a type system and automated type checker for verifying equivalences. While this tool is extremely efficient, it only covers a fixed set of cryptographic primitives (the same as Spec and Apte) and verifies an approximated equivalence, similar to the diff-equivalence. A completely different approach has been taken by Hirschi et al. [HBD16], identifying sufficient conditions provable by ProVerif for verifying unlinkability properties, implemented in the tool Ukano, a front-end to the ProVerif tool. Ukano does however not verify equivalence properties in general.

Contributions We significantly improve the theoretical understanding and the practical verification of equivalence when the number of protocol sessions is bounded. We emphasise that even in this setting, the system under study has an infinite state space due to the term algebra modelling cryptographic primitives. Our work targets the wide class of cryptographic primitives that can be represented by a subterm convergent rewriting system. Concretely, we provide

1. new tight complexity results for static equivalence ($\sim$), trace equivalence ($\approx_t$) and labelled bisimilarity ($\approx_\ell$);
2. a novel procedure for deciding trace equivalence and labelled bisimilarity for the class of cryptographic primitives modelled by a destructor subterm convergent rewrite system;
3. an implementation of our procedure for trace equivalence in a new tool called Deepsec (DEciding Equivalence Properties for SECurity protocols).

We detail the three contributions below.

Complexity. We provide the first complexity results for deciding trace equivalence and labelled bisimilarity in the applied pi calculus, without any restriction on the class of protocols (other than bounding the number of sessions). In particular, our results are not restricted to determinate processes, allow for else branches and do not approximate equivalence. Let us also highlight one small, yet substantial difference with existing work: we do not consider cryptographic primitives (rewrite systems) as constants of the problem. As most modern verification tools allow for user-specified primitives [BSC16, SMCB13, SEMM14, CCCK16], our approach seems to better fit this reality. Typically, all existing procedures for static equivalence can only be claimed PTIME because of this difference and are actually exponential in the sizes of signature or equational theory. Our complexity results are summarised in fig. 1. All our lower bounds hold for subterm convergent rewrite systems (destructor or not) and even for the positive fragment (without else branches). *En passant*, we present results for the pi calculus: although investigated in [BT00], complexity was unknown when restricted to a bounded number of sessions. Still, our main result is the coNEXP completeness (and in particular, the decidability) of trace equivalence and labelled bisimilarity for destructor subterm convergent rewrite systems.

Decision procedure. We present a novel procedure based on a symbolic semantics and constraint solving. Unlike most other work, our procedure decides equivalences *exactly*, i.e. without approximations. Moreover, it does not restrict the class of processes (except for replication), nor the use of else branches, and is correct for any cryptographic primitives that can be modelled by a subterm convergent destructor rewrite system (see section 2). The design of the procedure did greatly benefit from our complexity study, and was developed in order to obtain tight complexity upper bounds.

	Pure pi calculus	Applied pi calculus (destr.) subterm convergent
$\sim$	LOGSPACE	coNP complete
$\approx_t$	Π_2 complete	coNEXP complete
$\approx_\ell$	PSPACE complete	coNEXP complete

Figure 1: Summary of complexity results.

Tool implementation. We implemented our procedure for trace equivalence in a new tool, DEEPSEC. While still a prototype, DEEPSEC was carefully engineered. The tool output is available in pretty printed html format and allows to step through an attack, if any is found. DEEPSEC can also distribute the computation, thus exploiting multicore architectures or clusters of computers to their fullest. Finally, we integrated several classical optimisations for trace-equivalence analysis, e.g. *partial order reductions* (POR) [BDH15]. This has appeared to reduce the search space dramatically, making the tool scale well in practice despite the high theoretical complexity (coNEXP).

Through extensive benchmarks, we compare DEEPSEC to other tools limited to a bounded number of protocol sessions: APTE, SPEC, AKISS and SAT-EQUIV. Our tool is significantly more efficient—by several orders of magnitude—than APTE, SPEC and AKISS, even though DEEPSEC covers a strictly larger class of protocols than APTE and SPEC. Besides, its performances are comparable to SAT-EQUIV, which still outperforms DEEPSEC when the number of parallel processes significantly increase. This gap in performances seems unavoidable as DEEPSEC operates on a much larger class of protocols (more primitives, else branches, no limitation to simple processes, termination guaranteed).

Part of the benchmarks consists of classical authentication protocols and focuses on demonstrating scalability of the tool when augmenting the number of parallel protocol sessions. The other examples include more complex protocols, such as Abadi and Fournet's anonymous authentication protocol [AF04], the protocols implemented in the European passport [For04], the AKA protocol used in 3G mobile telephony, as well as the Prêt-à-Voter [RS06] and the Helios [Adi08] e-voting protocols.

Contents

2 Model

2.1 Messages and cryptographic primitives

We model the messages that are exchanged as first-order terms over a given signature and we equip them with a term rewrite system.

Term signature. We assume an infinite set of *names*, denoted $\mathcal{N} = \{a, b, \ldots, \}$ which are used to model atomic data. We partition $\mathcal{N}$ into two disjoint infinite sets $\mathcal{N}_{\mathsf{pub}}$ and $\mathcal{N}_{\mathsf{prv}}$. $\mathcal{N}_{\mathsf{pub}}$ is used to model *public* names that are known by the adversary, or generated by the adversary. $\mathcal{N}_{\mathsf{prv}}$ is the set of *private* names, used to model secret keys and nonces. We also define an infinite set of *variables*, denoted $\mathcal{X}$ as well as a *signature* $\mathcal{F} = \{\mathsf{f}/n, \mathsf{g}/m, \ldots\}$, that is a finite set of *function symbols* with their *arity*. More precisely, we consider $\mathcal{F} = \mathcal{F}_c \uplus \mathcal{F}_d$ where $\mathcal{F}_c$ is the set of *constructors* and $\mathcal{F}_d$ is the set of *destructors*. Intuitively, in contrast to constructors, the destructors represent cryptographic primitives that can fail during their execution, which may for instance happen when incorrect arguments were given to them (*e.g.* decryption may fail when one tries to decrypt a cipher text using the wrong key).

Example 1. A signature for modelling standard cryptographic primitives (symmetric and asymmetric encryption, concatenation and hash) is $\mathcal{F} = \mathcal{F}_c \cup \mathcal{F}_d$ where $\mathcal{F}_c$ and $\mathcal{F}_d$ are defined as follows:

$$\mathcal{F}_c = \{\mathsf{senc}/2,\ \mathsf{aenc}/2,\ \mathsf{pk}/1,\ \langle\,\rangle/2,\ \mathsf{h}/1\}$$
$$\mathcal{F}_d = \{\mathsf{sdec}/2,\ \mathsf{adec}/2,\ \mathsf{proj}_1/1,\ \mathsf{proj}_2/1\}$$

The function aenc (resp. senc) represents asymmetric (resp. symmetric) encryption with corresponding decryption function adec (resp. sdec) and public key pk. Concatenation is represented by $\langle\,\rangle$ with associated projectors proj_1 and proj_2. h represents the hash function.

Terms and substitutions *Terms* are defined as names, variables, and function symbols applied to other terms. For any $\mathsf{F} \subseteq \mathcal{F}$, $\mathsf{N} \subseteq \mathcal{N}$ and $\mathsf{V} \subseteq \mathcal{X}$, the set of terms built from N and V by applying function symbols in F is denoted by $\mathcal{T}(\mathsf{F}, \mathsf{N} \cup \mathsf{V})$. We denote by $vars(t)$ and $names(t)$ the sets of variables and names respectively occurring in a term t. We denote by $st(t)$ the set of subterms of u, and by $sst(t)$ its strict subterms, i.e. $sst(t) = st(t) \setminus \{t\}$. We write $root(t)$ for the function symbol, name, variable or constant that is at the root position of the term t. A *constructor term* is a term belonging to $\mathcal{T}(\mathcal{F}_c, \mathcal{N} \cup \mathcal{X})$.Given a term t we denote by $|t|$ its size, i.e., the number of nodes of the syntax tree of the term. When stating some of our complexity results we suppose a succinct representation of terms as DAGs with maximal sharing (which may be exponentially more concise). We define $|t|_{\mathsf{dag}} = |st(t)|$ to be the DAG size of t, which equals the number of different subterms occurring in t (as the DAG with maximal sharing has exactly one node for each subterm). These notions of size are naturally lifted to sets and sequences of terms (where the sharing of DAGs is common to all elements of the set or sequence).

We distinguish two types of terms: *protocol terms* and *recipes*. For this we refine the set of variables $\mathcal{X}$ to be composed of three different types of variables and define $\mathcal{X} = \mathcal{X}^1 \uplus \mathcal{X}^2 \uplus \mathcal{AX}$ where $\mathcal{X}^1 = \{x, y, z, \ldots\}$ is the set of *first-order variables*, $\mathcal{X}^2 = \{X, Y, \ldots\}$ is the set of *second-order variables* and $\mathcal{AX} = \{\mathsf{ax}_1, \ldots, \mathsf{ax}_n, \ldots\}$ is the set of *axioms*. We also write $\mathcal{AX}_i$ for the set $\{\mathsf{ax}_1, \ldots, \mathsf{ax}_i\}$ and by convention define $\mathcal{AX}_0 = \emptyset$. We define $vars^1(t) = vars(t) \cap \mathcal{X}^1$, $vars^2(t) = vars(t) \cap \mathcal{X}^2$ and $axioms(t) = vars(t) \cap \mathcal{AX}$ to denote the set of first-order variables, the set of second-order variables and the set of axioms occurring in t, respectively. We further refine the set of second-order variables as $\mathcal{X}^2 = \bigcup_{i \geq 0} \mathcal{X}^2_i$, such that for any $i \geq 0$ we have that $(\bigcup_{j<i} \mathcal{X}^2_j) \subset \mathcal{X}^2_i$. We moreover define $\mathcal{X}^2_{:i}$ to be $(\mathcal{X}^2_i \setminus \mathcal{X}^2_{i-1})$ and write $X{:}i$ when we want to emphasise that $X \in \mathcal{X}^2_{:i}$. Intuitively, a second order variable $X{:}i$ may only be instantiated with terms whose axioms are in $\mathcal{AX}_i$ and second-order variables in $\mathcal{X}^2_i$.

Protocol terms are the set of terms in $\mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X}^1)$. A ground protocol term t is a *message* if $u\!\downarrow$ is a constructor term for all $u \in st(t)$. We define the predicate $Msg(\cdot)$ on terms such that $Msg(t)$ is true if and only if t is a message.

Recipes are defined as the set of terms in $\mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{X}^2 \cup \mathcal{AX})$. They represent the actions of the attacker and in particular how he deduces messages. Axioms can be seen as pointers to the hypotheses (either initially known messages or protocol outputs). A ground recipe ξ is a recipe that does not contain second-order variables, i.e., $vars^2(\xi) = \emptyset$. Note that a ground recipe may however contain axioms. We also define by $\mathcal{T}^2_i = \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{X}^2_i \cup \mathcal{AX}_i)$, i.e., the set of recipes that may only contain axioms and second-order variables up to i.

Substitutions are mappings from variables to terms. We usually denote a substitution σ as $\{x_1 \mapsto u_1; \ldots; x_n \mapsto u_n\}$. The *domain* of σ, denoted $dom(\sigma)$, is the set $\{x_1, \ldots, x_n\}$ and the *image* of σ, denoted $img(\sigma)$, is the set $\{u_1, \ldots, u_n\}$. We denote the substitution whose domain is empty by $\emptyset$. We moreover write $|\sigma|$ for $|dom(\sigma)|$, we denote by $\sigma|_X$ the substitution whose domain is restricted to variables in X and write $\sigma' \succeq \sigma$ when σ' extends σ, i.e., $\sigma'|_{dom(\sigma)} = \sigma$. As usual we homomorphically extend substitutions to apply to (sets of, sequences of) terms and use postfix notation for application, i.e., we write $t\sigma$ for $\sigma(t)$. We suppose that substitutions will preserve the types of the terms: they will map first-order variables to protocol terms, axioms to protocol terms and second-order variables to recipes.

Remark 1. We suppose that substitutions preserve types: variables of $\mathcal{X}^1 \cup \mathcal{AX}$ are mapped to protocol terms and variables of $\mathcal{X}^2_i$ to recipes of $\mathcal{T}^2_i$.

Rewriting and unification Behaviour of primitives are modelled through a finite convergent *rewriting system* $\mathcal{R}$ defined on $\mathcal{F}$ such that for all $(\ell, r) \in \mathcal{R}$, usually denoted $\ell \to r \in \mathcal{R}$, $\ell \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $r \in \mathcal{T}(\mathcal{F}_c, \mathcal{X})$. A term u is reduced into v by $\mathcal{R}$, denoted $u \to_{\mathcal{R}} v$, if there exist a substitution σ, a position p of t and $(\ell, r) \in \mathcal{R}$ such that $u|_p = \ell\sigma$ and $u[r\sigma]_p = v$. We denote by $\to^*_{\mathcal{R}}$ the transitive closure of $\to_{\mathcal{R}}$. Since $\mathcal{R}$ is convergent, we denote by $u\!\downarrow_{\mathcal{R}}$ the (unique) normal form of u, *i.e.* $u \to^*_{\mathcal{R}} u\!\downarrow_{\mathcal{R}}$ and no reduction step is possible on $u\!\downarrow_{\mathcal{R}}$. To lighten notation, whenever $\mathcal{R}$ is clear from the context, we omit $\mathcal{R}$ and simply write $u \to v$ and $u\!\downarrow$.

Definition 1 (rewriting system). *A rewriting system $\mathcal{R}$ is*

- subterm *if for all $\ell \to r \in \mathcal{R}$, r is either a strict subterm of ℓ or a ground term in normal form;*
- destructor *if for all $\ell \to r \in \mathcal{R}$, ℓ is a term $\mathsf{g}(u_1, \ldots, u_n)$ where the u_i's are constructor terms and $\mathsf{g} \in \mathcal{F}_d$.*

Example 2. The signature in example 1 can typically be equipped with the rewriting system:

$$\mathsf{sdec}(\mathsf{senc}(x,y),y) \to x \quad \mathsf{proj}_1(\langle x,y\rangle) \to x \quad \mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y)),y) \to x \quad \mathsf{proj}_2(\langle x,y\rangle) \to y$$

Two terms t and u are (syntactically) *unifiable* if there exists a substitution σ such that $t\sigma = u\sigma$. A substitution σ is said to be a *most general unifier* of t and u if for any unifier θ of t

and u, there exists τ such that $\theta = \sigma\tau$. It is well known that for any two unifiable terms s and t, there is a unique most general unifier, up to variable renaming. We denote this most general unifier by $mgu(s,t)$. We assume w.l.o.g. that $dom(mgu(s,t)) \cap vars(img(mgu(s,t))) = \emptyset$ and $dom(mgu(s,t)) \cup vars(img(mgu(s,t))) \subseteq vars(s,t)$. We also define the set of unifiers modulo a rewrite system $\mathcal{R}$.

Definition 2. *Let $\{s_i, t_i\}_{i \in I}$ be a set of pairs of terms and $X = vars(\{s_i, t_i\}_{i\in I})$. A set of substitutions $mgu_{\mathcal{R}}(\{s_i \overset{?}{=} t_i\}_{i \in I})$ is called a* complete set of unifiers modulo $\mathcal{R}$ *of the system of equations $\{s_i =^? t_i\}_{i\in I}$ if each of the following holds:*

1. *$dom(\sigma) \subseteq vars(X)$ for each $\sigma \in mgu_{\mathcal{R}}(\{s_i =^? t_i\}_{i \in I})$;*
2. *$s_i\sigma{\downarrow} = t_i\sigma{\downarrow}$, $Msg(s\sigma)$ and $Msg(t\sigma)$ for each $i \in I$ and each $\sigma \in mgu_{\mathcal{R}}(\{s_i =^? t_i\}_{i\in I})$;*
3. *for any substitution θ such that $s_i\theta =_{\mathcal{R}} t_i\theta$ for every $i \in I$, there exists a substitution $\sigma \in mgu_{\mathcal{R}}(\{s_i =^? t_i\}_{i\in I},)$ and a substitution τ with $\theta[X]{\downarrow} = \sigma\tau|_X{\downarrow}$.*

For singleton systems, we also write $mgu_{\mathcal{R}}(s,t)$ instead of $mgu_{\mathcal{R}}(\{s =^? t\})$. Such complete sets of unifiers can be computed for (destructor) subterm convergent rewrite systems using narrowing. Also note that for destructor rewrite systems we require that the unifiers only yield messages and not arbitrary terms.

Example 3. Using the rewrite system of example 2, if $a \in \mathcal{N}$ and $t = \mathsf{sdec}(\mathsf{proj}_1(x), y)$ we have $mgu_{\mathcal{R}}(a,t) = \{\{x \mapsto \langle \mathsf{senc}(a,y), z\rangle\}\}$. An other, less usual, example consist of unifying $t = \mathsf{proj}_1(x)$ with itself modulo $\mathcal{R}$ since we do not get the identity but $mgu_{\mathcal{R}}(t,t) = \{\{x \mapsto \langle y, z\rangle\}\}$.

2.2 Processes

Plain processes, the core of the modelling of distributed protocols, are defined by the following grammar by following in the steps of [AF01]:

$$\begin{array}{rlr} P, Q := & 0 & \text{null} \\ & P \mid Q & \text{parallel} \\ & \mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q & \text{conditional} \\ & \overline{u}\langle v\rangle.P & \text{output} \\ & u(x).P & \text{input} \end{array}$$

where u, v are terms and x is a first-order variable. Nested parallel processes may be written without parenthesis assuming a right-associative convention:

$$\prod_{i=1}^{n} P_i \ \triangleq\ P_1 \mid \cdots \mid P_n \ \triangleq\ P_1 \mid (P_2 \mid (\cdots \mid P_n) \cdots)$$

Inputs bind variables and define their scope: a plain process is then *closed* if it does not have free variable, i.e. all variables are bound. We say that a plain process P is *positive* when each of the conditionals of P is of the form $\mathsf{if}\ u = v\ \mathsf{then}\ Q\ \mathsf{else}\ 0$ (written $\mathsf{if}\ u = v\ \mathsf{then}\ Q$). We denote by $|P|$ the size of the process, that is the size of the process tree plus the size of the terms (in explicit representation) and by $|P|_{DAG}$ its size when terms are represented by DAGs (with global sharing).

Remark 2. The calculus of [AF01] has a replication operator requiring a mechanism (name restriction) to handle unbounded private-name generation. Here, private names are simply those of $\mathcal{N}_{\mathsf{prv}}$.

Example 4. Consider the protocol for private authentication [AF04], described informally using Alice-and-Bob notation:

$$\begin{array}{rll} X \to B : & \mathsf{aenc}(\langle N_X, \mathsf{pk}(sk_X)\rangle, \mathsf{pk}(sk_B)) & \\ B \to X : & \mathsf{aenc}(\langle N_X, \langle N_B, \mathsf{pk}(sk_B)\rangle\rangle, \mathsf{pk}(sk_A)) & \text{if X=A} \\ & \mathsf{aenc}(N_B, \mathsf{pk}(sk_B)) & \text{otherwise} \end{array}$$

B accepts authentication requests from A but not from other parties. However, the protocol should hide to any outsider that B is willing to engage with A—which explains the decoy message sent when B is contacted by a different party. The role of B can be specified in the applied pi calculus, writing $t = \mathsf{adec}(x, sk_B)$, $t_1 = \mathsf{proj}_1(t)$ and $t_2 = \mathsf{proj}_2(t)$:

$$\begin{array}{rl} B = & c(x). \\ & \mathsf{if}\ t_2 = \mathsf{pk}(sk_A)\ \mathsf{then} \\ & \quad \bar{c}\langle \mathsf{aenc}(\langle t_1, \langle N_B, \mathsf{pk}(sk_B)\rangle\rangle, \mathsf{pk}(sk_A))\rangle \\ & \mathsf{else}\ \bar{c}\langle \mathsf{aenc}(N_B, \mathsf{pk}(sk_B))\rangle \end{array}$$

where $sk_A, sk_B, N_A, N_B \in \mathcal{N}_{\mathsf{prv}}$, $c \in \mathcal{N}_{\mathsf{pub}}$. *Anonymity* can be stated as equivalence of B and $B' = B\{sk_A \mapsto sk_{A'}\}$, assuming that the attacker has access to all public keys involved. Indeed, this means that an attacker sees no difference between B willing to engage with A or A'.

Next we define *extended processes* which intuitively represent the set of plain processes simultaneously executed in parallel, together with the knowledge aggregated by the attacker along the run through public outputs:

Definition 3 (extended process). *An extended process is a pair $A = (\mathcal{P}, \Phi)$ where:*

- $\Phi = \{\mathsf{ax}_1 \mapsto u_1, \ldots, \mathsf{ax}_n \mapsto u_n\}$ *is a substitution of domain $dom(\Phi) \subseteq \mathcal{AX}$ and range $img(\Phi) \subseteq \mathcal{T}(\mathcal{F}_c, \mathcal{N})$;*
- $\mathcal{P}$ *is a multiset of closed plain processes.*

Moreover, we require that all variables are bound only once in $\mathcal{P}$, i.e., a variable x can appear only in one input of $\mathcal{P}$.

The substitution Φ is called the *frame*. The frame contains the attacker's initial knowledge by recording the protocol outputs. It will be used to evaluate recipes, *e.g.* $\xi\Phi$ represents the message obtained by application of Φ on the recipe ξ. Given an extended process A, we denote the frame of A by $\Phi(A)$. Finally, we use a double-bracket notation $\{\!\{\cdot\}\!\}$ for multisets.

2.3 Semantics

We define the operational semantics of extended processes through a labelled transition relation $A \xrightarrow{a}_{\mathsf{c}}^{\mathcal{R}} B$ where A, B are extended processes and a is an *action* and $\mathcal{R}$ a rewrite system. For the sake of readability we often omit to specify $\mathcal{R}$, as it is generally clear from the context. The alphabet of actions $\mathcal{A}$ consists of

- the set of input actions $\xi(\zeta)$ where ξ is a recipe for the channel, and ζ a recipe for the term to be input;
- the set of output actions $\overline{\xi}\langle \mathsf{ax}_n\rangle$ where ξ is a recipe for the channel, and ax_n is the axiom added to the frame, pointing to the output term.

In a transition $A \xrightarrow{a}_{\mathsf{c}} B$, a is either the empty word ε or a letter from $\mathcal{A}$. The transition relation is defined by the rules given in fig. 2. Note that we check that all terms that have to be evaluated during the execution are messages.

$$(\mathcal{P} \cup \{\!\{0\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P}, \Phi) \qquad \text{(NULL)}$$

$$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P\}\!\}, \Phi) \qquad \text{(THEN)}$$
$$\text{if } \mathit{Msg}(u),\ \mathit{Msg}(v) \text{ and } u{\downarrow} = v{\downarrow}$$

$$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{Q\}\!\}, \Phi) \qquad \text{(ELSE)}$$
$$\text{if either } \neg\mathit{Msg}(u),\ \neg\mathit{Msg}(v) \text{ or } u{\downarrow} \neq v{\downarrow}$$

$$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P, v(x).Q\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P, Q\{x \mapsto t\}\}\!\}, \Phi) \qquad \text{(COMM)}$$
$$\text{if } \mathit{Msg}(u),\ \mathit{Msg}(v),\ \mathit{Msg}(t) \text{ and } u{\downarrow} = v{\downarrow}$$

$$(\mathcal{P} \cup \{\!\{P \mid Q\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P, Q\}\!\}, \Phi) \qquad \text{(PAR)}$$

$$(\mathcal{P} \cup \{\!\{u(x).P\}\!\}, \Phi) \xrightarrow{\xi(\zeta)}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P\{x \mapsto \zeta\Phi{\downarrow}\}\}\!\}, \Phi) \qquad \text{(IN)}$$
$$\text{if } \xi, \zeta \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathit{dom}(\Phi)),\ \mathit{Msg}(u),\ \mathit{Msg}(\xi\Phi),\ \mathit{Msg}(\zeta\Phi) \text{ and } \xi\Phi{\downarrow} = u{\downarrow}$$

$$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P\}\!\}, \Phi) \xrightarrow{\overline{\xi}\langle \mathsf{ax}_n\rangle}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P\}\!\}, \Phi \cup \{\mathsf{ax}_n \mapsto t{\downarrow}\}) \qquad \text{(OUT)}$$
$$\text{if } \xi \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathit{dom}(\Phi)),\ \mathit{Msg}(u),\ \mathit{Msg}(\xi\Phi),\ \mathit{Msg}(t),\ \xi\Phi{\downarrow} = u{\downarrow}, \text{ and } n = |\Phi| + 1$$

Figure 2: Semantics of the calculus (w.r.t. an implicit rewrite system $\mathcal{R}$)

Given a word $w \in \mathcal{A}^*$, we denote the relation $A \overset{w}{\Rightarrow}_{\mathsf{c}} B$ when $A \xrightarrow{a_1}_{\mathsf{c}} \ldots \xrightarrow{a_n}_{\mathsf{c}} B$ with $w = a_1 \ldots a_n$. The process calculus is very similar to the replication-free fragment of the original applied π-calculus [AF01] and has the same expressiveness. Finally, taking $\mathcal{F} = \emptyset$ and $\mathcal{R} = \emptyset$, i.e., the only terms are names and variables, we retrieve a calculus that is similar to the finite, pure π-calculus [MPW92].

Example 5. We introduce a toy example which will serve as a pedagogical running example. It was designed for its simplicity and capacity to illustrate the different notions defined in this paper. If $b \in \{0, 1\} \subseteq \mathcal{N}_{\mathsf{pub}}$ and $c \in \mathcal{N}_{\mathsf{pub}}$:

$$\begin{aligned} P^b &\triangleq c(x).\,\mathsf{if}\ \mathsf{proj}_2(x) = b\ \mathsf{then}\ \overline{c}\langle 0\rangle\ \mathsf{else}\ \overline{c}\langle \mathsf{proj}_2(x)\rangle \\ Q &\triangleq c(x).\,\overline{c}\langle \mathsf{proj}_2(x)\rangle \end{aligned}$$

Process Q forwards the second component of a term received through a public channel c. P^0 and P^1 have a similar behaviour as Q except that on input values $\langle t, 1\rangle$, P^1 outputs 0 rather than 1. We illustrate the semantics on P^1, e.g. by forwarding a hash $\mathsf{h}(n)$, $n \in \mathcal{N}_{\mathsf{pub}}$, sent by the attacker:

$$\begin{aligned} &(\{\!\{P^1\}\!\}, \emptyset) \\ &\xrightarrow{c(\langle 0, \mathsf{h}(n)\rangle)}_{\mathsf{c}} \left(\left\{\!\!\left\{ \begin{array}{l} \mathsf{if}\ \mathsf{proj}_2(\langle 0, \mathsf{h}(n)\rangle) = 1\ \mathsf{then}\ \overline{c}\langle 0\rangle \\ \mathsf{else}\ \overline{c}\langle \mathsf{proj}_2(\langle 0, \mathsf{h}(n)\rangle)\rangle \end{array} \right\}\!\!\right\}, \emptyset \right) \\ &\overset{\overline{c}\langle \mathsf{ax}_1\rangle}{\Longrightarrow}_{\mathsf{c}} (\emptyset, \{\mathsf{ax}_1 \mapsto \mathsf{h}(n)\}) \end{aligned}$$

2.4 Equivalences

Equivalences between two processes intuitively express the fact that no attacker can distinguish these two processes. They can be used to formalise many interesting security properties, such as anonymity, unlinkability, strong secrecy, *etc.* We focus in this section on three equivalences: *static equivalence*, *labelled bisimilarity* and *trace equivalence.*

2.4.1 Static equivalence

Static equivalence expresses that two sequences of terms, represented by frames, are indistinguishable to an attacker. This notion has been extensively studied (see e.g. [AC06]). Intuitively, two frames are in static equivalence if an attacker cannot distinguish them, even when applying arbitrary primitives to the messages in the frames.

Definition 4 (static equivalence). *Let Φ and Φ' two closed frames and $\mathcal{R}$ a rewrite system. We say that Φ and Φ' are* statically equivalent*, written $\Phi \sim^{\mathcal{R}} \Phi'$, when $dom(\Phi) = dom(\Phi')$ and when for all ground recipes ξ, ζ, if $axioms(\xi, \zeta) \subseteq dom(\Phi)$ then:*

- *$Msg(\xi\Phi)$ if and only if $Msg(\xi\Phi')$*
- *if $Msg(\xi\Phi)$ and $Msg(\zeta\Phi)$ then $\xi\Phi{\downarrow_{\mathcal{R}}} = \zeta\Phi{\downarrow_{\mathcal{R}}}$ if and only if $\xi\Phi'{\downarrow_{\mathcal{R}}} = \zeta\Phi'{\downarrow_{\mathcal{R}}}$.*

We also lift static equivalence to extended processes: $A \sim^{\mathcal{R}} B$ iff $\Phi(A) \sim^{\mathcal{R}} \Phi(B)$.

The first item of definition 4 states that the failure of a destructor on one frame also yield a failure on the other frame. The second item is more classical and is used to verify that an equality holding on one frame also hold on the other frame.

Example 6. One informal property of encryption schemes is that an encrypted message should look unintelligible, which is expressed by the static equivalence of $\Phi_0 = \{\mathsf{ax}_1 \mapsto \mathsf{senc}(m, k)\}$ and $\Phi_1 = \{\mathsf{ax}_1 \mapsto m'\}$ for all $m \in \mathcal{N}_{\mathsf{pub}}$ and $k, m' \in \mathcal{N}_{\mathsf{prv}}$. This is not the case anymore once the key is revealed: $\Phi_0 \cup \{\mathsf{ax}_2 \mapsto k\} \not\sim \Phi_1 \cup \{\mathsf{ax}_2 \mapsto k\}$ (try recipes $\xi = \mathsf{sdec}(\mathsf{ax}_1, \mathsf{ax}_2)$ and $\zeta = m$).

2.4.2 Trace equivalence

Trace equivalence is a generalisation of static equivalence to the case where an attacker actively interacts with the protocol. Intuitively, two extended processes A and B are in trace equivalence if any sequence of actions of A can be matched by the same sequence of actions in B such that the resulting frames are statically equivalent.

Definition 5 (trace equivalence). *Let A and B be two extended processes and $\mathcal{R}$ a rewrite system. We say that $A \sqsubseteq_t^{\mathcal{R}} B$ if for all $A \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (P, \Phi)$, there exists (P', Φ') such that $B \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (P', \Phi')$ and $\Phi \sim^{\mathcal{R}} \Phi'$. We say that A and B are* trace equivalent*, denoted $A \approx_t^{\mathcal{R}} B$, when $A \sqsubseteq_t^{\mathcal{R}} B$ and $B \sqsubseteq_t^{\mathcal{R}} A$.*

One can easily prove that trace inclusion $\sqsubseteq_t^{\mathcal{R}}$ is a (partial) ordering on extended processes and that $\approx_t^{\mathcal{R}}$ is an equivalence relation.

2.4.3 Labelled bisimilarity

The last equivalence states that an external observer cannot make a difference between two protocols, taking into account not only the frames but also the dynamic behaviour of the processes.

Definition 6 (labelled bisimilarity)**.** *Let $\mathcal{R}$ a rewrite system. Labelled bisimilarity ($\approx_\ell^{\mathcal{R}}$) is the largest symmetric relation such that for all extended processes A and B, $A \approx_\ell^{\mathcal{R}} B$ entails:*

1. *$A \sim^{\mathcal{R}} B$; and*
2. *if $A \xrightarrow{\alpha}_{\mathsf{c}} A'$ then there exists B' such that $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$ and $A' \approx_\ell^{\mathcal{R}} B'$.*

One can easily prove that this defines an equivalence relation on extended processes. In [AF01] it was shown that labelled bisimilarity coincides with observational equivalence, a barbed congruence relation.

Example 7. Consider the processes introduced in example 5. The informal statement that P^0 and Q have the same behaviour is expressed by the equivalence $(P^0, \emptyset) \approx_\ell (Q, \emptyset)$. On the contrary $(P^1, \emptyset) \not\approx_t (Q, \emptyset)$ since the transition $(P^1, \emptyset) \xRightarrow{c(1).\bar{c}\langle \mathsf{ax}_1 \rangle}_{\mathsf{c}} \{\mathsf{ax}_1 \mapsto \mathsf{0}\}$ cannot be matched in $(Q, \emptyset)$.

2.5 Complexity and decision problems

In this section we shortly remind some background about complexity, mainly introducing our notations and define the decision problems related to the previously introduced equivalences.

2.5.1 Basic complexity classes

Given $f : \mathbb{N} \to \mathbb{N}$, we define $\texttt{TIME}(f(n))$ (resp. $\texttt{SPACE}(f(n))$) the class of problems decidable by a deterministic Turing machine running in time (resp. in space) at most $f(n)$. It is common to define the following classes:

$$\begin{aligned} \textsc{logspace} &= \bigcup_{p \in \mathbb{N}} \texttt{SPACE}(\log(n^p)) & \textsc{ptime} &= \bigcup_{p \in \mathbb{N}} \texttt{TIME}(n^p) \\ \textsc{pspace} &= \bigcup_{p \in \mathbb{N}} \texttt{SPACE}(n^p) & \textsc{exptime} &= \bigcup_{p \in \mathbb{N}} \texttt{TIME}(2^{n^p}) \end{aligned}$$

One can define their non-deterministic counterparts NLOGSPACE, NPTIME, NPSPACE and NEXPTIME. Given a (non-deterministic) class $\mathcal{C}$, we call co-$\mathcal{C}$ the class of problems whose negation is in $\mathcal{C}$. From now on, we often omit the suffix TIME in the name of time-complexity classes for the sake of succinctness, and write L instead of LOGSPACE. Then it is known that:

$$\textsc{l} \subseteq \textsc{nl}{=}\textsc{conl} \subseteq \textsc{p} \subseteq \textsc{np},\textsc{conp} \subseteq \textsc{pspace}{=}\textsc{npspace} \subseteq \textsc{exp} \subseteq \textsc{nexp},\textsc{conexp}$$

By abuse of notation, we identify a class $\mathcal{C}$ of decisions problems and the class $F\mathcal{C}$ of functions (from string to string, in terms of Turing machines) computable within the resources of $\mathcal{C}$. We use *many-to-one* polynomial-time reductions to define complete problems for complexity classes above PTIME, and mention the more general notion of *oracle* reduction, deciding a problem with a constant-time black box for an other problem. The class of problems decidable in $\mathcal{C}$ with an oracle for a problem Q is noted $\mathcal{C}^Q$. When Q is complete for a class $\mathcal{D}$ w.r.t. a notion of reduction executable in $\mathcal{C}$, we may write $\mathcal{C}^{\mathcal{D}}$ instead.

2.5.2 Polynomial hierarchy

In terms of alternating Turing machines, the difference between NP and PSPACE lies in their capacity to express (quantifier) alternation. However, the gap between "purely existential" (NP) and "unrestricted alternation" (PSPACE) seems a little harsh. The polynomial hierarchy defines the intermediate classes.

Definition 7 (polynomial hierarchy)**.** *The polynomial hierarchy* PH *consists of the classes* Σ_n *defined by* $\Sigma_0 =$P *and* $\Sigma_i = \text{NP}^{\Sigma_{i-1}}$ *if* $i > 0$*. In particular,* $\Sigma_1 = \text{NP}$*. We also write* Π_i *for* coΣ_i*.*

2.5.3 Decision problems for equivalences

For each of the previously mentioned equivalences, whenever the rewrite system is clear from the context we omit $\mathcal{R}$ and simply write $\sim$, $\approx_t$ and $\approx_\ell$. We can now define the decision problems associated to each of these equivalences.

Definition 8 (parameterised equivalence problem)**.** *Let* $\mathcal{F}$ *be a signature,* $\mathcal{R}$ *a rewriting system defined on* $\mathcal{F}$ *and* $\asymp \in \{\sim, \approx_t, \approx_\ell\}$*. We define the decision problem* $\mathsf{Equiv}^{\asymp}_{\mathcal{R},\mathcal{F}}$*:*

> INPUT*: two extended processes* A, B *defined on* $\mathcal{F}$*.*
>
> QUESTION*:* $A \asymp^{\mathcal{R}} B$*.*

In [AC06], it was first shown that for an arbitrary convergent rewriting system $\mathcal{R}$, and signature $\mathcal{F}$ the problem $\mathsf{Equiv}^{\approx}_{\mathcal{R},\mathcal{F}}$ is undecidable, but for any subterm convergent rewriting system $\mathcal{R}$, $\mathsf{Equiv}^{\approx}_{\mathcal{R},\mathcal{F}}$ can be solved in polynomial time. We note that in $\mathsf{Equiv}^{\asymp}_{\mathcal{R},\mathcal{F}}$, $\mathcal{R}$ and $\mathcal{F}$ are not part of the input and that all previously proposed procedures in [AC06, CDK12, CBC10] are actually exponential in $\mathcal{R}$ or $\mathcal{F}$. Moreover, this definition does not allow to give a lower bound for the whole class of subterm convergent rewriting system (the lower bounds may be different for particular rewriting systems). We therefore define an alternate problem which takes $\mathcal{R}$ and $\mathcal{F}$ as additional inputs. Given the undecidability of the general case we need to restrict the class of rewrite systems, or processes. We do that by the means of a predicate ψ which must hold on the input.

Definition 9 (general equivalence problem)**.** *Let* $\asymp \in \{\sim, \approx_t, \approx_\ell\}$*. We define the decision problem* $\mathsf{Equiv}^{\asymp}_{\psi}$*:*

> INPUT*: a signature* $\mathcal{F}$*, a rewriting system* $\mathcal{R}$ *defined on* $\mathcal{F}$*, two extended processes* A, B *defined on* $\mathcal{F}$ *such that* $\psi(\mathcal{F}, \mathcal{R}, A, B)$ *holds.*
>
> QUESTION*:* $\mathsf{Equiv}^{\asymp}_{\mathcal{R},\mathcal{F}}(A, B)$*.*

To lighten notation we will often just write

"$\mathsf{Equiv}^{\asymp}$ for subterm convergent rewrite systems and positive processes"

rather than

"$\mathsf{Equiv}^{\asymp}_{\psi}$ where $\psi(\mathcal{F}, \mathcal{R}, A, B)$ holds when $\mathcal{R}$ is a subterm convergent rewrite systems and A, B are positive processes".

Remark 3. It is important to be precise on whether terms in the input are provided as trees or DAGs (as the DAG representation may be exponentially more concise). In this paper, when proving complexity lower bounds we suppose that terms are represented as trees; when showing upper bounds we suppose that terms are encoded as DAGs. As terms in tree representation can be converted into DAGs in polynomial time (in the tree size), the presented results also hold if the other encoding would have been chosen.

Part I
Complexity lower bounds

3 Tools for proving complexity lower bounds

In this section we introduce a few tools and gadgets that will be helpful for encoding problems.

3.1 Internal non-deterministic choice

A classical feature of CCS which is not directly part of the applied π-calculus is *non-deterministic choice*. $P + Q$ can be executed either as P or as Q, which is formalized by the following rules:

$$(\mathcal{P} \cup \{\!\!\{P + Q\}\!\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\!\{P\}\!\!\}, \Phi) \tag{Choice-L}$$

$$(\mathcal{P} \cup \{\!\!\{P + Q\}\!\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\!\{Q\}\!\!\}, \Phi) \tag{Choice-R}$$

The classical encoding for $+$ within the initial calculus is formally defined by a process transformation $[\![\cdot]\!]$, where:

$$[\![P + Q]\!] \triangleq \overline{s}\langle s\rangle \mid s(x).[\![P]\!] \mid s(y).[\![Q]\!] \qquad \text{where } s \in \mathcal{N}_{\mathsf{prv}} \text{ and } x, y \in \mathcal{X}^1 \text{ are fresh} \tag{1}$$

and all other cases of the syntax are handled as homomorphic extensions of $[\![\cdot]\!]$. As for the parallel operator we will sometimes use the big operator $\sum$ assuming right-associativity. The correctness of this translation with respect to $\approx_t$ and $\approx_\ell$ will be stated later on in section 3.3.

Remark 4. By "fresh name" (or variable), we mean that the term does not appear in other parameters of the problem, should it be in processes, their translations, or frames.

We also introduce the $\mathtt{Choose}(x)$ construct which non-deterministically assigns either 0 or 1 to x. $\mathtt{Choose}(x).P$ silently reduces to either $P\{x \mapsto \mathsf{0}\}$ or $P\{x \mapsto \mathsf{1}\}$ and $\mathtt{Choose}(\vec{x}).P$ is defined as $\mathtt{Choose}(x_1).\mathtt{Choose}(x_2)\ldots\mathtt{Choose}(x_n).P$ where $\vec{x} = x_1, \cdots, x_n$. Formally, we extend the operational semantics with the rule

$$(\mathcal{P} \cup \{\!\!\{\mathtt{Choose}(x).P\}\!\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\!\{P\{x \mapsto \mathsf{0}\}\}\!\!\}, \Phi) \tag{Choose-0}$$

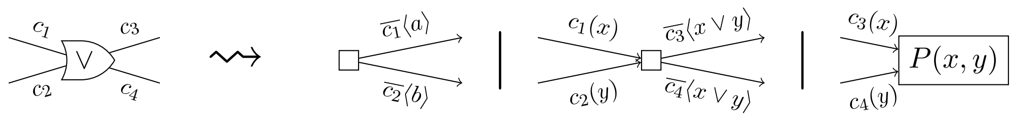
$$(\mathcal{P} \cup \{\!\!\{\mathtt{Choose}(x).P\}\!\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\!\{P\{x \mapsto \mathsf{1}\}\}\!\!\}, \Phi) \tag{Choose-1}$$

and define

$$[\![\mathtt{Choose}(y).P]\!] \triangleq (\overline{d}\langle 0\rangle + \overline{d}\langle 1\rangle) \mid d(y).[\![P]\!] \quad \text{with } d \in \mathcal{N}_{\mathsf{prv}} \text{ is fresh}$$

3.2 Boolean circuits and formulae

About circuits Complete problems in complexity theory often involve boolean formulae (e.g., SAT or QBF). The ability to evaluate boolean formulae, or boolean circuits in general, within the applied π-calculus is therefore curcial. We can implement such a feature by the means of private channels and internal communication: each edge of a boolean circuit Γ indeed mimics a channel transmitting a boolean over a network (fig. 3).



c_1
c_3
c_2
c_4
$\vee$
$\overline{c_1}\langle a\rangle$
$\overline{c_2}\langle b\rangle$
$c_1(x)$
$c_2(y)$
$\overline{c_3}\langle x\vee y\rangle$
$\overline{c_4}\langle x\vee y\rangle$
$c_3(x)$
$c_4(y)$
$P(x,y)$

Figure 3: Simulation of an OR-gate within the applied π-calculus

Formally, the essence of circuits lies in so-called *logical gates* which are boolean functions with at most two inputs. We assume without loss of generality that the gate has at most two (identical) outputs, to be given as input to other gates. Logical gates usually range over the constants $\mathsf{0}$ and $\mathsf{1}$ and the predicates $\wedge$, $\vee$ and $\neg$ with the usual truth tables but we may use other common operators such as $=$. From that a *boolean circuit* is an acyclic graph of logical gates: each input (resp. output) of a gate is either isolated or connected to a unique output (resp. input) of an other gate, which defines the edges of this graph.

Such a circuit Γ with m isolated inputs and p isolated outputs thus models a boolean function $\Gamma : \mathbb{B}^m \to \mathbb{B}^n$ (where $\mathbb{B} = \{\mathsf{0},\mathsf{1}\}$). We write $(c_1, c_2, g, c_3, c_4) \in \Gamma$ to state that $g : \mathbb{B}^2 \to \mathbb{B}$ is a gate of Γ whose inputs are passed through edges c_1 and c_2 and whose output is sent to edges c_3 and c_4. This notation is naturally lifted to other in-outdegrees.

Embedding into the calculus The syntax of plain processes is now extended with the construction $x_1, \cdots, x_n \leftarrow \Gamma(b_1, \cdots, b_m).P$ where $\Gamma : \mathbb{B}^m \to \mathbb{B}^n$ is a circuit, $x_1, \ldots, x_n$ variables and $b_1, \cdots, b_m$ terms. We fix two distinct terms $\mathsf{0}, \mathsf{1} \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}})$ to model $\mathbb{B}$ within the calculus, and the labelled operational semantics is extended with the rule:

$$(\mathcal{P} \cup \{\!\{\vec{x} \leftarrow \Gamma(\vec{b}).P\}\!\}, \Phi) \xrightarrow{\varepsilon}_{\mathsf{c}} (\mathcal{P} \cup \{\!\{P\{\vec{x} \mapsto \Gamma(\vec{b}{\downarrow})\}\}\!\}, \Phi) \qquad (\textsc{Valuate})$$
$$\text{if } \mathit{Msg}(\vec{b}) \text{ and } \vec{b}{\downarrow} \subseteq \mathbb{B}$$

Now we have to extend the definition of $[\![\cdot]\!]$ (previous subsection) to handle the new operator. For simplicity we only consider the case where gates have two inputs and two outputs: handling lower arities is straightforward. If $(c_1, c_2, g, c_3, c_4) \in \Gamma$, we first define:

$$[\![c_1, c_2, g, c_3, c_4]\!] \triangleq c_1(x).c_2(y). \prod_{b,b' \in \mathbb{B}} \mathsf{if}\ x = b\ \mathsf{then\ if}\ y = b'\ \mathsf{then}\ (\overline{c_3}\langle g(b,b')\rangle \mid \overline{c_4}\langle g(b,b')\rangle)$$

where $c_1, c_2, c_3, c_4 \in \mathcal{N}_{\mathsf{prv}}$ (assuming that different circuits in a process do not share edges). To sum it up, we simply see circuit edges as private channels and simulate the logical flow of the gate. It is then easily extended:

$$[\![\vec{x} \leftarrow \Gamma(\vec{b}).P]\!] \triangleq \prod_{k=1}^{m} \overline{c_{i_k}}\langle b_k\rangle \ \mid \prod_{(c_1,c_2,g,c_3,c_4)\in\Gamma} [\![c_1, c_2, g, c_3, c_4]\!] \ \mid\ c_{o_1}(x_1) \ldots c_{o_n}(x_n).[\![P]\!] \qquad (2)$$

where $(c_{i_k})_{k=1}^m$ (resp. $(c_{o_k})_{k=1}^n$) are the isolated input (resp. output) edges of Γ.

Remark 5. When b and b' are fixed booleans, $g(b, b')$ denotes the boolean obtained from the truth table of g: we emphasise that g is *not* a function symbol of the signature $\mathcal{F}$.

Remark 6. We assume that every input of a circuit goes through at least one gate and has at least one output. This is to avoid irrelevant side cases in proofs.

3.3 Correctness of the translation

Now we dispose of an extended syntax and semantics as well as a mapping $[\![\cdot]\!]$ removing the new constructors from a process. The correctness of this translation is proven in appendix A.2:

Lemma 1. *Let $\approx_t^+$ and $\approx_\ell^+$ be the notions of trace equivalence and labelled bisimilarity over the extended calculus (the flag $^+$ being omitted outside of this lemma). For all extended processes $A = (\mathcal{P}, \Phi)$, the translation $[\![A]\!] = ([\![\mathcal{P}]\!], \Phi) = (\{\!\{[\![P]\!] \mid P \in \mathcal{P}\}\!\}, \Phi)$ can be computed in polynomial time, $A \approx_t^+ [\![A]\!]$ and $A \approx_\ell^+ [\![A]\!]$.*

Remark 7. As the finite, pure, and positive fragments of the applied π-calculus are closed under $[\![\cdot]\!]$, sums and circuits can be safely used within any intersection of such fragments.

4 Complexity lower bounds for the pure π-calculus

In this section we give lower bounds for the complexity of $\mathsf{Equiv}^{\approx_t}$ (resp. $\mathsf{Equiv}^{\approx_\ell}$) for the *positive* pure π-calculus, i.e., taking $\mathcal{F} = \mathcal{R} = \emptyset$. We show that deciding trace equivalence, respectively labelled bisimilarity, is Π_2-hard, resp. PSPACE-hard, in this setting. This is done by reduction from QSAT$_2$ (resp. QBF) to non-equivalence.

4.1 Trace equivalence

To show that trace equivalence is Π_2-hard we proceed by a reduction from QSAT$_2$ to non-equivalence. As QSAT$_i$ is known to be Σ_i-complete [Pap03] this allows us to conclude that trace equivalence is Π_2-hard.

Definition 10 (QSAT$_i$)**.** *For any $i \in \mathbb{N}$, the problem QSAT$_i$ is defined as follows:*

> INPUT*: A boolean formula φ and a partition $X_1, \cdots, X_i$ of its variables.*
> QUESTION*: Does the statement $\exists X_1, \forall X_2, \cdots, QX_i, \varphi$ hold, where $Q = \exists$ if i is odd and $Q = \forall$ otherwise?*

Let φ be a boolean formula whose variables are partitioned into $\{\vec{x}\} \cup \{\vec{y}\}$. Our goal is to construct two processes A and B such that:

$$A \not\approx_t B \text{ if and only if } \exists\vec{x}.\forall\vec{y}.\varphi(\vec{x},\vec{y}) = 1 \tag{3}$$

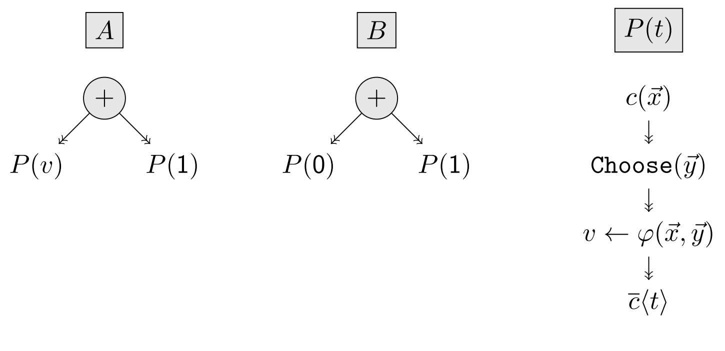


Figure 4: Schematic definition of A and B

Consider three distinct names $c, \mathsf{0}, \mathsf{1} \in \mathcal{N}_{\mathsf{pub}}$ (the last two modelling booleans for the circuit construction of section 3.2). Processes A and B are depicted in fig. 4. Intuitively, the process $P(t)$ (where t is a term which may depend on the variables bound by P) gets a valuation of $\vec{x}$ from the attacker, internally chooses a valuation of $\vec{y}$, computes the value of $\varphi(\vec{x}, \vec{y})$ using rule VALUATE, and outputs t. From that it is quite easy to see that A and B have the same set of traces **iff** for all valuation of $\vec{x}$, there exists a valuation of $\vec{y}$ such that $\varphi(\vec{x}, \vec{y}) = \mathsf{0}$. More formally, we define $P(t)$, A and B as follows.

$$\begin{aligned} P(t) &\triangleq c(\vec{x}).\ \mathtt{Choose}(\vec{y}).\ v \leftarrow \varphi(\vec{x}, \vec{y}).\ \bar{c}\langle t\rangle \\ A &\triangleq P(v) + P(\mathsf{1}) \\ B &\triangleq P(\mathsf{0}) + P(\mathsf{1}) \end{aligned}$$

The correctness of the reduction, that is to say eq. (3), is formally proven in the long version, appendix A.3.

Theorem 1. $\mathsf{Equiv}_{\emptyset,\emptyset}^{\approx_t}$ *is* Π_2*-hard on positive processes.*

4.2 Labelled bisimilarity

To prove that labelled bisimilarity is PSPACE-hard for the positive pure pi calculus we proceed by reduction from satisfiability of Quantified Boolean Formulae (QBF), which is PSPACE-complete [Pap03].

Lemma 2 (QBF)**.** *The following problem is* PSPACE*-complete:*

> INPUT*: A boolean formula φ with variables $x_1, \dots, x_n, y_1, \dots, y_n$.*
> QUESTION*: Does $\exists x_1, \forall y_1, \dots, \exists x_n, \forall y_n, \varphi$ hold?*

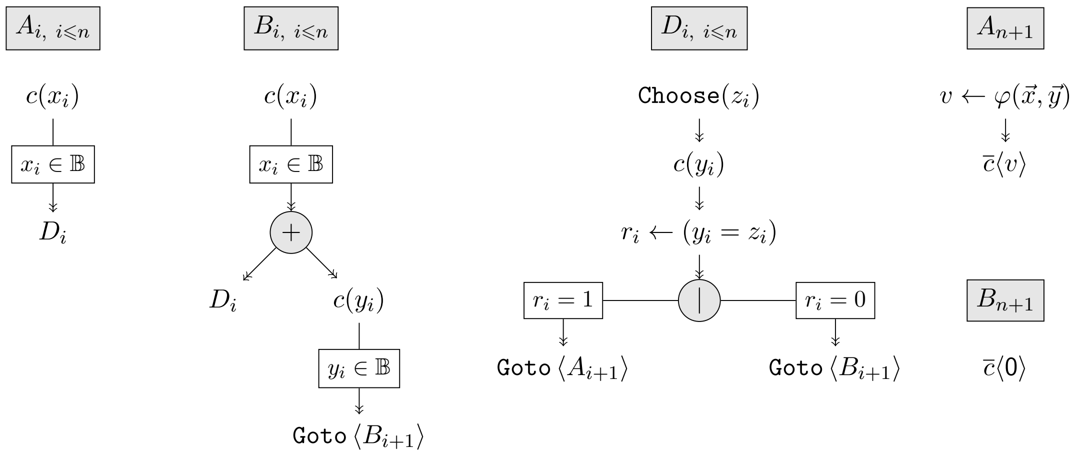
The case of labelled bisimilarity is more involved as QBF allows arbitrary quantifier alternation. Let φ be a boolean formula whose variables are partitioned into $\{x_1, \dots, x_n\} \cup \{y_1, \dots, y_n\}$ for some $n \in \mathbb{N}$. We construct (in polynomial time in the size of φ and n) two processes A and B such that:

$$A \not\approx_\ell B \ \text{ if and only if } \ \exists x_1 \forall y_1 \dots \exists x_n \forall y_n.\ \varphi(x_1, \dots, x_n, y_1, \dots, y_n) = 1 \tag{4}$$

Both QBF and labelled bisimilarity may be seen as bisimulation games: an attacker plays the $\exists$-quantifiers (selects a transition in a process) whereas a defender responds with the $\forall$-quantifiers (tries to find a similarly-labelled sequence of transitions in the other process). The role of A and B is to implement this intuitive connection: the attacker moves will be simulated by public inputs $c(x_i)$ and the defender responses by instructions $\mathtt{Choose}(z_i).c(y_i)$. The structure of A and B is then designed to constrain the moves of the two players so that the winning condition of the attacker is exactly $\exists x_1 \forall y_1 \dots \exists x_n \forall y_n.\ \varphi(\vec{x}, \vec{y}) = 1$.

A and B are defined inductively by processes A_i, B_i and D_i, depicted in fig. 5, structured in a way that, in a bisimulation game:

1. the attacker chooses the instance of x_i;
2. the defender chooses the instance of z_i and can force the attacker to instanciate y_i with the same value (the attacker not doing so allows for a trivial victory of the defender).



$A_{i,\ i\leqslant n}$
$c(x_i)$
$x_i \in \mathbb{B}$
D_i
$B_{i,\ i\leqslant n}$
$c(x_i)$
$x_i \in \mathbb{B}$
$+$
D_i
$c(y_i)$
$y_i \in \mathbb{B}$
$\texttt{Goto}\ \langle B_{i+1}\rangle$
$D_{i,\ i\leqslant n}$
$\texttt{Choose}(z_i)$
$c(y_i)$
$r_i \leftarrow (y_i = z_i)$
$r_i = 1$
$|$
$r_i = 0$
$\texttt{Goto}\ \langle A_{i+1}\rangle$
$\texttt{Goto}\ \langle B_{i+1}\rangle$
A_{n+1}
$v \leftarrow \varphi(\vec{x}, \vec{y})$
$\overline{c}\langle v\rangle$
B_{n+1}
$\overline{c}\langle 0\rangle$

Figure 5: Schematic definition of A_i and B_i

We fix a family of private channels $(c_P)_P \subseteq \mathcal{N}_{\mathsf{prv}}$ indexed by processes P which will be used to simulate instructions $\texttt{Goto}\ \langle P\rangle$. We use a shortcut $\overline{d}\langle \overrightarrow{t}^{p}\rangle$ for an indexed sequence of terms $(t_i)_i$ to denote the sequence of outputs:

$$\overline{d}\langle \overrightarrow{t}^{p}\rangle \triangleq \overline{d}\langle t_1\rangle \ldots \overline{d}\langle t_p\rangle$$

and a similar notation for sequences of inputs. Then the $\texttt{Goto}$ feature is implemented as follows, allowing for passing and receiving program states through parallel processes:

$$\texttt{Goto}\ \langle A_i\rangle \triangleq \overline{c_{A_i}}\langle \overrightarrow{x}^{i}, \overrightarrow{y}^{i}\rangle \qquad \texttt{Goto}\ \langle B_i\rangle \triangleq \overline{c_{B_i}}\langle \overrightarrow{x}^{i}, \overrightarrow{y}^{i}\rangle$$
$$\texttt{GetEnv}\ \langle A_i\rangle .P \triangleq c_{A_i}(\overrightarrow{x}^{i}, \overrightarrow{y}^{i}) \qquad \texttt{GetEnv}\ \langle B_i\rangle .P \triangleq c_{B_i}(\overrightarrow{x}^{i}, \overrightarrow{y}^{i})$$

Formally the processes A_i, B_i and D_i are defined below. We stress out that A and B are closed (as required) but that A_i, B_i and D_i are not in general. Fixing a public channel $c \in \mathcal{N}_{\mathsf{pub}}$, we write:

$$\begin{aligned}
\forall i \leqslant n,\ A_i &\triangleq c(x_i).\ x_i \leftarrow x_i.\ D_i \\
\forall i \leqslant n,\ B_i &\triangleq c(x_i).\ x_i \leftarrow x_i.\ (D_i + (c(y_i).\ y_i \leftarrow y_i.\ \texttt{Goto}\ \langle B_{i+1}\rangle)) \\
A_{n+1} &\triangleq v \leftarrow \varphi(\vec{x}, \vec{y}).\ \overline{c}\langle v\rangle \\
B_{n+1} &\triangleq \overline{c}\langle 0\rangle \\
D_i &\triangleq \texttt{Choose}(z_i).\ c(y_i).\ r_i \leftarrow (y_i = z_i). \\
&\quad ((\mathsf{if}\ \ r_i = 1\ \ \mathsf{then}\ \ \texttt{Goto}\ \langle A_{i+1}\rangle) \\
&\qquad \mid (\mathsf{if}\ \ r_i = 0\ \ \mathsf{then}\ \ c(y_i).\ y_i \leftarrow y_i.\ \texttt{Goto}\ \langle B_{i+1}\rangle))
\end{aligned}$$

As in the reduction for trace equivalence (section 4.1), the $\texttt{Choose}(\alpha)$ simulates non-deterministic choice among $\mathbb{B}$; the construction $\alpha \leftarrow \alpha$, which may seem useless, encodes the test $\alpha \in \mathbb{B}$. Finally, we define A and B by putting the auxiliary processes in parallel and connecting the $\texttt{Goto}$'s to the $\texttt{getEnv}$'s:

$$A \triangleq A_1 \mid C \qquad B \triangleq B_1 \mid C \qquad C \triangleq \prod_{i=2}^{n+1}(\texttt{GetEnv}\ \langle A_i\rangle .A_i) \ \mid\ \prod_{i=2}^{n+1}(\texttt{GetEnv}\ \langle B_i\rangle .B_i)$$

A and B can be computed in time $\mathcal{O}(n^2+|\varphi|)$ in a straightforward way. The last ingredient of our reduction, namely the proof of eq. (4), follows the steps of the intuition presented above; the technical proof itself may be found in appendix A.3 in the long version.

Theorem 2. $\mathsf{Equiv}^{\approx_\ell}_{\emptyset,\emptyset}$ *is* PSPACE*-hard on positive processes.*

5 Complexity lower bounds for the applied π-calculus

5.1 Static equivalence

The problem of static equivalence for subterm convergent equational theories has first been studied by [AC06]. It is proven PTIME *provided that the rewrite system is a constant of the problem.* All procedures for static equivalence in the literature can also be claimed PTIME only when the rewrite system is fixed. We prove here that the problem is actually coNP hard when the rewrite system is part of the input.

Reduction By reduction from SAT, let $\varphi = \bigwedge_{i=1}^{p} C_i$ a boolean formula in CNF with n variables $x_1, \ldots, x_n$ and p clauses $C_1, \ldots, C_p$. Then we define the signature $\mathcal{F} = \mathcal{F}_c \uplus \mathcal{F}_d$:

$$\mathcal{F}_c = \{\mathsf{0},\ \mathsf{1},\ \mathsf{f}/2,\ \mathsf{g}/2\} \qquad\qquad \mathcal{F}_d = \{\mathsf{eval}/n\}$$

which is equipped with the rewrite system $\mathcal{R}$ defined by the following $p+1$ rules:

$$\mathsf{eval}(\mathsf{f}(x_1, y), \ldots, \mathsf{f}(x_n, y)) \to \mathsf{0} \qquad\qquad \mathsf{eval}(\mathsf{g}(t^i_1, y), \ldots, \mathsf{g}(t^i_n, y)) \to \mathsf{0}$$

where $1 \leqslant i \leqslant p$ and

$$t^i_j = \begin{cases} x_j & \text{if } x_j \text{ does not appear in } C_i \\ \mathsf{0} & \text{if } x_j \text{ appears positively in } C_i \\ \mathsf{1} & \text{if } x_j \text{ appears negatively in } C_i \end{cases}$$

Note that this definition assumes that no clause of φ contains both a litteral and its negation. Such clauses are tautological and can be removed by a preprocessing step in LOGSPACE, inducing no loss of generality. Intuitively, if $t_1, \ldots, t_n \in \{\mathsf{0}, \mathsf{1}\}$, $\mathsf{eval}(\mathsf{g}(t_1, y), \ldots, \mathsf{g}(t_n, y))$ is a message and reduces to $\mathsf{0}$ **iff** the valuation $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ falsifies φ. Note also that $\mathcal{R}$ is convergent, subterm and destructor. Then the reduction lies in the fact that, for some $k \in \mathcal{N}_{\mathsf{prv}}$, $\{\mathsf{f}(\mathsf{0}, k),\ \mathsf{f}(\mathsf{1}, k)\} \sim \{\mathsf{g}(\mathsf{0}, k),\ \mathsf{g}(\mathsf{1}, k)\}$ **iff** φ is unsatisfiable.

Theorem 3. $\mathsf{Equiv}^{\sim}_{\mathcal{F},\mathcal{R}}$ *for $\mathcal{R}$ subterm destructor is co*NP *hard.*

5.2 Reducing succinct satisfiability to positive equivalences

In this section, we show a coNEXP lower bound for trace equivalence and labelled bismilarity by co-reduction from SUCCINCT 3SAT. A *succinct instance of* 3SAT is a boolean circuit $\Gamma : \{\mathsf{0}, \mathsf{1}\}^{m+2} \to \{\mathsf{0}, \mathsf{1}\}^{n+1}$ (we refer to the definitions of section 3.2). Through binary representation of integers, Γ is interpreted as a function $[\![\Gamma]\!] : [\![0, 2^m-1]\!] \times [\![1, 3]\!] \to \{\mathsf{0}, \mathsf{1}\} \times [\![0, 2^n-1]\!]$. This way, Γ encodes a CNF formula $[\![\Gamma]\!]_\varphi$ with 2^n variables $\vec{x} = x_0, \cdots, x_{2^n-1}$ and 2^m clauses:

$$[\![\Gamma]\!]_\varphi(\vec{x}) = \bigwedge_{i=0}^{2^m-1} \ell^1_i \vee \ell^2_i \vee \ell^3_i \quad \text{where} \begin{cases} \ell^j_i = x_k & \text{if } [\![\Gamma]\!](i,j) = (0,k) \\ \ell^j_i = \neg x_k & \text{if } [\![\Gamma]\!](i,j) = (1,k) \end{cases}$$

Rephrasing, $[\![\Gamma]\!](i,j)$ returns a sign bit and the j^{th} variable of the i^{th} clause of $[\![\Gamma]\!]_\varphi$. This induces the following question which is the typical complete problem for NEXP [Pap03].

Definition 11 (SUCCINCT 3SAT). *The* SUCCINCT 3SAT *problem is defined as follows:*

> INPUT*: A boolean circuit* Γ *with* $m+2$ *inputs and* $n+1$ *outputs.*
> QUESTION*: Is the* 3SAT*-formula* $[\![\Gamma]\!]_\varphi$ *satisfiable?*

Consider an instance of this problem, Γ, with $m+2$ inputs and $n+1$ outputs and we design $\mathcal{F}$, $\mathcal{R}$ subterm destructor and A and B positive processes such that $A \not\approx_t B$ **iff** $A \not\approx_\ell B$ **iff** $[\![\Gamma]\!]_\varphi$ is satisfiable.

Term algebra Terms are built over the following signature:

$$\begin{array}{rll}
\mathcal{F} \triangleq & 0,\ 1, & \text{(booleans } \mathbb{B}\text{)} \\
& \mathsf{Node}/2,\ \pi/2, & \text{(binary trees)} \\
& \mathsf{h}/2, & \text{(one-way binary hash)} \\
& \mathsf{h_N}/2,\ \mathsf{h}_\mathbb{B}/2,\ \mathsf{Test_N}/1,\ \mathsf{Test_B}/1 & \text{(testable binary hashes)}
\end{array}$$

We equip this term algebra with the rewriting system $\mathcal{R}$ containing the following rules modelling subtree extraction (for binary trees) and argument testing (for hashes):

$$\begin{array}{ccc}
\pi(\mathsf{Node}(x,y),0) \to x & \pi(\mathsf{Node}(x,y),1) \to y & \\
\mathsf{Test_N}(\mathsf{h_N}(\mathsf{Node}(x,y),z)) \to 1 & \mathsf{Test_B}(\mathsf{h}_\mathbb{B}(0,z)) \to 1 & \mathsf{Test_B}(\mathsf{h}_\mathbb{B}(1,z)) \to 1
\end{array}$$

In particular $\mathcal{R}$ is subterm and destructor, the destructor symbols being π, $\mathsf{Test_N}$ and $\mathsf{Test_B}$. We will also use a shortcut for recursive subtree extraction: if $\ell \in \mathcal{T}(\mathcal{F}, \mathcal{N} \cup \mathcal{X}^1)^\star$ is a finite sequence of protocol terms, the notation $t_{|\ell}$ is inductively defined by:

$$t_{|\varepsilon} \triangleq t \qquad\qquad t_{|b\cdot\ell} \triangleq \pi(t,b)_{|\ell}$$

Core of the reduction Recall that we are studying a formula in CNF $[\![\Gamma]\!]_\varphi$ with 2^n variables and 2^m clauses. In particular, given a valuation of its 2^n variables, we can verify in non-deterministic polynomial time in n, m that it falsifies $[\![\Gamma]\!]_\varphi$:

1. guess an integer $i \in [\![0, 2^m-1]\!]$ as a sequence of m bits;
2. obtain the three literals of the i^{th} clause of $[\![\Gamma]\!]_\varphi$ (requiring three runs of the circuit Γ) and verify that the valuation falsifies the disjunction of the three literals.

This non-deterministic verification is the essence our reduction. In the actual processes:

- a process `CheckTree`(x) checks that x is a correct encoding of a valuation, that is, that x is a complete binary tree of height n whose leaves are booleans;
- a process `CheckSat`(x) implements the points *1.* and *2.* above.

All of this is then formulated as equivalence properties within A and B (see the intermediary lemmas in the next paragraph for details). Intuitively, we want to express the following statement by equivalence properties: "*for all term x, either x is not an encoding of a valuation or falsifies a clause of $[\![\Gamma]\!]_\varphi$*". A schematized definition is proposed in fig. 6.

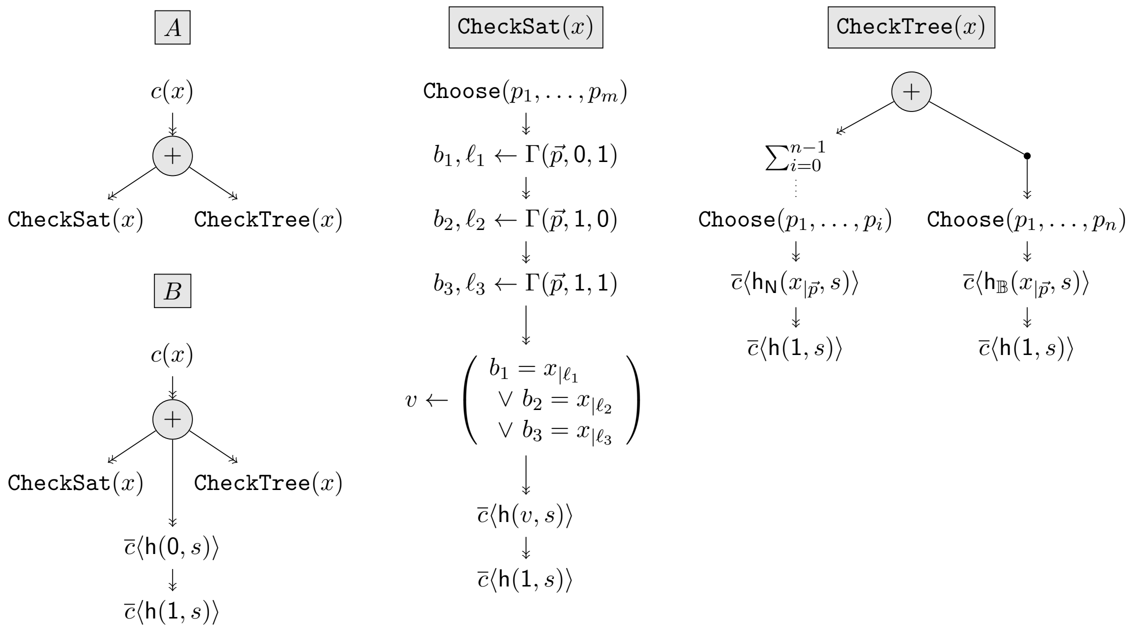

A
$c(x)$
$+$
$\texttt{CheckSat}(x)$
$\texttt{CheckTree}(x)$
B
$c(x)$
$+$
$\texttt{CheckSat}(x)$
$\texttt{CheckTree}(x)$
$\overline{c}\langle \mathsf{h}(0,s)\rangle$
$\overline{c}\langle \mathsf{h}(1,s)\rangle$
$\texttt{CheckSat}(x)$
$\texttt{Choose}(p_1,\dots,p_m)$
$b_1,\ell_1 \leftarrow \Gamma(\vec{p},0,1)$
$b_2,\ell_2 \leftarrow \Gamma(\vec{p},1,0)$
$b_3,\ell_3 \leftarrow \Gamma(\vec{p},1,1)$
$v \leftarrow \begin{pmatrix} b_1 = x_{\mid \ell_1} \\ \vee\ b_2 = x_{\mid \ell_2} \\ \vee\ b_3 = x_{\mid \ell_3} \end{pmatrix}$
$\overline{c}\langle \mathsf{h}(v,s)\rangle$
$\overline{c}\langle \mathsf{h}(1,s)\rangle$
$\texttt{CheckTree}(x)$
$+$
$\sum_{i=0}^{n-1}$
$\texttt{Choose}(p_1,\dots,p_i)$
$\overline{c}\langle \mathsf{h}_{\mathsf{N}}(x_{\mid \vec{p}},s)\rangle$
$\overline{c}\langle \mathsf{h}(1,s)\rangle$
$\texttt{Choose}(p_1,\dots,p_n)$
$\overline{c}\langle \mathsf{h}_{\mathbb{B}}(x_{\mid \vec{p}},s)\rangle$
$\overline{c}\langle \mathsf{h}(1,s)\rangle$

Figure 6: Informal definition of A and B

Formal construction Let us now define the processes depicted in fig. 6 properly; note that all the proofs about the correctness of this construction are relegated to appendix A.4.2 but we still state several intermediary lemmas in order to highlight the proof structure. But first of all, let us give a name to a frame which is at the core of our reduction:

$$\Phi_0 = \{\mathsf{ax}_1 \mapsto \mathsf{h}(0,s),\ \mathsf{ax}_2 \mapsto \mathsf{h}(1,s)\}$$

Φ_0 is reached after executing the central branch of B and everything is about knowing under which conditions a frame statically equivalent to Φ_0 can be reached in A. Let us define the processes themselves now. We fix $s \in \mathcal{N}_{\mathsf{prv}}$ and define, if x is a protocol term:

$$\begin{aligned}\texttt{CheckTree}(x) \triangleq \sum_{i=0}^{n-1} \big(\ &\texttt{Choose}(p_1,\dots,p_i).\ \overline{c}\langle \mathsf{h}_{\mathsf{N}}(x_{\mid p_1\cdots p_i},s)\rangle.\ \overline{c}\langle \mathsf{h}(1,s)\rangle\ \big)\\ &+ \texttt{Choose}(p_1,\dots,p_n).\ \overline{c}\langle \mathsf{h}_{\mathbb{B}}(x_{\mid p_1\cdots p_n},s)\rangle.\ \overline{c}\langle \mathsf{h}(1,s)\rangle\end{aligned}$$

Lemma 3. *Let x be a message which is not complete binary tree of height n with boolean leaves. Then there exists a reduction* $\texttt{CheckTree}(x) \overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} (\{\!\{P\}\!\},\emptyset)$ *such that* $(\{\!\{P\}\!\},\emptyset) \approx_\ell (\overline{c}\langle \mathsf{h}(0,s)\rangle.\ \overline{c}\langle \mathsf{h}(1,s)\rangle,\emptyset)$.

Now let us move on to $\texttt{CheckSat}(x)$. This process binds a lot of variables:

- $\vec{p} = p_1,\dots,p_m$ models the non-deterministic choice of a clause number in $[\![0,2^m-1]\!]$;
- b_i,ℓ_i, $i \in [\![1,3]\!]$, where ℓ_i is a sequence of n variables, model the literals of the clause chosen above (b_i is the negation bit and ℓ_i the identifier of the variable);
- v stores whether the chosen clause is satisfied by the valuation modelled by x.

$$\begin{aligned}\texttt{CheckSat}(x) \triangleq\ & \texttt{Choose}(\vec{p}).\\ & b_1, \ell_1 \leftarrow \Gamma(\vec{p}, 0, 1).\\ & b_2, \ell_2 \leftarrow \Gamma(\vec{p}, 1, 0).\\ & b_3, \ell_3 \leftarrow \Gamma(\vec{p}, 1, 1).\\ & v \leftarrow (b_1 = x_{|\ell_1} \vee\ b_2 = x_{|\ell_2} \vee\ b_3 = x_{|\ell_3}).\\ & \overline{c}\langle \mathsf{h}(v, s)\rangle.\overline{c}\langle \mathsf{h}(1, s)\rangle\end{aligned}$$

Lemma 4. *Let x be a complete binary tree of height n whose leaves are booleans, and val_x be the valuation mapping the variable number $i = \sum_{k=1}^{m} p_k 2^{k-1}$ of $[\![\Gamma]\!]_\varphi$ to $x_{|p_1 \cdots p_m} \in \mathbb{B}$. Then, if val_x does not satisfy $[\![\Gamma]\!]_\varphi$ then there exists $\texttt{CheckSat}(x) \overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} (\{\!\{P\}\!\}, \emptyset)$ such that $(\{\!\{P\}\!\}, \emptyset) \approx_\ell (\overline{c}\langle \mathsf{h}(0, s)\rangle. \overline{c}\langle \mathsf{h}(1, s)\rangle, \emptyset)$.*

We can finally wrap up everything by defining A and B and stating the last part of the correctness theorem. We recall that all the proofs can be found in appendix A.4.2.

$$\begin{aligned} A &\triangleq c(x).(\texttt{CheckSat}(x)\ +\ \texttt{CheckTree}(x))\\ B &\triangleq c(x).(\texttt{CheckSat}(x)\ +\ \texttt{CheckTree}(x)\ +\ \overline{c}\langle \mathsf{h}(0, s)\rangle.\overline{c}\langle \mathsf{h}(1, s)\rangle)\end{aligned}$$

Lemma 5. *$[\![\Gamma]\!]_\varphi$ is satisfiable* **iff** *$A \not\approx_t B$* **iff** *$A \not\approx_\ell B$.*

Theorem 4. *$\mathsf{Equiv}^{\approx_t}_{\mathcal{F},\mathcal{R}}$ and $\mathsf{Equiv}^{\approx_\ell}_{\mathcal{F},\mathcal{R}}$ for $\mathcal{R}$ subterm destructor are* co*NEXP-hard on positive processes.*

Part II
Complexity upper bounds

6 A symbolic setting

In this section we introduce a symbolic process calculus: rather than requiring the attacker to perform concrete actions, we record the constraints that need to hold on the inputs provided by the adversary. This representation provides a finite representation of the infinite set of actions available to the attacker. In this section we first define *constraint systems* which record the constraints on adversary inputs. Next we introduce the notion of *most general solutions*, a notion reminiscent to the set of most general unifiers, but for solutions of constraint system. Then we introduce a symbolic process calculus, allowing to reason on non ground processes and associated constraint systems. Finally, we define a *partition tree*: a partition tree may be seen as the tree of all symbolic executions of two initial symbolic processes, but additionally partitioning the solutions such that all solutions in a same node yield statically equivalent processes. The partition tree conceptually captures all information needed to decide trace equivalence and observational equivalence and will be used in the next section to provide upper bounds on the complexity. Its construction is delayed to Section part III.

6.1 Constraint systems

Before defining constraint systems themselves we introduce a first-order logic for expressing the constraints. We introduce two kinds of atomic formulas:

- $\xi \vdash^? u$, and
- $u =^? v$

where u and v are constructor protocol terms and ξ is a recipe such that $root(\xi) \notin \mathcal{F}_c$. $\xi \vdash^? u$ is called a *deduction fact.* A positive literal $u =^? v$ is called an *equation* and a negative literal $\neg(u =^? v)$, which we also denote by $u \neq^? v$, is called a *disequation.*

A formula is a first-order logic formula built over atomic formulas. Given a formula F we denote by $vars^1(F)$, respectively $vars^2(F)$, the set of *free* first-order, respectively second-order, variables of F. A valuation of a formula F is a triple of substitutions (Φ, Σ, σ) where $\Phi : \mathcal{AX} \to \mathcal{T}(\mathcal{F}, \mathcal{N})$, $\Sigma : \mathcal{X}^2 \to \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{AX})$, $\sigma : \mathcal{X}^1 \to \mathcal{T}(\mathcal{F}_c, \mathcal{N})$ and that is grounding for F. The satisfaction relation $\models$ for atomic formulas is defined as

$$\begin{array}{rcl}(\Phi,\Sigma,\sigma) \models \xi \vdash^? u & \text{iff} & \xi\Sigma\Phi{\downarrow} = u\sigma{\downarrow} \text{ and } \mathit{Msg}(\xi\Sigma\Phi)\\ (\Phi,\Sigma,\sigma) \models (u =^? v) & \text{iff} & u\sigma = v\sigma\end{array}$$

and lifted to first-order formulas as usual. When a formula F does not contain any deduction constraint we sometimes simply write $\sigma \models F$ as the satisfaction of F does not depend on ϕ and Σ. When C is a conjunction of atomic formulas, we sometimes, by abuse of notation, interpret it as a set of formulas and write $(u =^?_{\mathcal{R}} v) \in \mathsf{C}$ when there exists C' such that $\mathsf{C} = (u =^?_{\mathcal{R}} v) \wedge \mathsf{C}'$.

We can now define *constraint systems.*

Definition 12. *A constraint system is a tuple* $(\Phi, \mathsf{D}, \mathsf{E}^1)$ *where:*

- $\Phi = \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\}$ *is a frame where* $t_1, \ldots, t_n$ *are constructor protocol terms;*
- D *is a conjunction of simple deducibility facts* $X \vdash^? t$ *where* $X \in \mathcal{X}^2_n$*, and* t *is a constructor protocol term;*
- E^1 *is a conjunction of formulas of the form* $u =^? v$ *or* $\forall y_1. \ldots. \forall y_k. \bigvee_{j=1}^{p} u_j \neq^? v_j$ *where* u_j, v_j, u, v *are constructor protocol terms for all* $j = 1, \ldots, p$*, and* $y_1, \ldots, y_k$ *are first order variables.*

Moreover, we assume that the following conditions hold:

- *second-order variables occur at most once in* D*, i.e., for all* $(X \vdash^? u), (Y \vdash^? v) \in \mathsf{D}$ *if* $u \neq v$ *then* $X \neq Y$*;*
- *for all* $k \in \{1, \ldots, n\}$*,* $x \in vars^1(t_k)$*, there exists* $X \in \mathcal{X}^2_{k-1}$ *such that* $(X \vdash^? x) \in \mathsf{D}$*;*
- $vars^1(\mathsf{E}^1) \subseteq vars^1(\mathsf{D})$*.*

We denote by $\mathcal{C}_\emptyset$ *the empty constraint system, i.e.* Φ *is the identity and* $\mathsf{D} = \mathsf{E}^1 = \top$*.*

Intuitively, a constraint system records constraints on a symbolic execution as follows. Φ corresponds to the frame. An input occurring after i outputs generates a deduction fact $X{:}i \vdash^? x$. Conditionals generate equations, respectively disequations, according to whether the symbolic trace chooses the "then" branch or the "else" branch. The second condition on constraint systems is generally called the origination property, and intuitively requires that any variable used in an output must have been input before. Looking ahead, the particular form of the disequalities can be explained by the fact that we use most general unifiers modulo $\mathcal{R}$ to transform the condition $u{\downarrow} \neq v{\downarrow}$ into a syntactic constraint using the following lemma.

Lemma 6. *Let* u, v *be two protocol terms.*

$$u\sigma{\downarrow} \neq v\sigma{\downarrow} \vee \neg Msg(u\sigma) \vee \neg Msg(v\sigma) \quad \Leftrightarrow \quad \sigma \vDash syn_{\mathcal{R}}(u \neq v)$$

where

- $syn_{\mathcal{R}}(u \neq v) = \bigwedge_{\delta \in mgu_{\mathcal{R}}(u,v)} \forall \tilde{z}_\delta. \bigvee_{x \in vars^1(u,v)} x \neq^? x\delta$*, and*
- $\tilde{z}_\delta = vars^1(u\delta, v\delta) \smallsetminus vars^1(u, v)$*.*

Given a constraint system $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1)$, we denote by $\Phi(\mathcal{C})$, $\mathsf{D}(\mathcal{C})$ and $\mathsf{E}^1(\mathcal{C})$ the corresponding elements of the tuple. The *structure* of a constraint system $\mathcal{C}$ is defined as $|\Phi(\mathcal{C})|$ and $\{X \mid (X \vdash^? x) \in \mathsf{D}(\mathcal{C})\}$.

Definition 13. *We say that* (Σ, σ) *is a solution of a constraint system* $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1)$ *when*

- $dom(\Sigma) = vars^2(\mathsf{D})$ *and* $dom(\sigma) = vars^1(\mathsf{D})$*,*
- $(\Phi\sigma, \Sigma, \sigma) \models \mathsf{D} \wedge \mathsf{E}^1$*.*

The substitution σ *is called the first-order solution and* Σ *the second-order solution of* $\mathcal{C}$*. The set of solutions of an initial constraint system* $\mathcal{C}$ *is denoted* $Sol(\mathcal{C})$*. A constraint system* $\mathcal{C}$ *is satisfiable if* $Sol(C) \neq \emptyset$*.*

Remark 8. Given a solution $(\Sigma, \sigma) \in \mathit{Sol}(C)$, due to the origination property, the second-order solution Σ uniquely defines the first-order solution as $\sigma = \sigma_n$ where

- $\mathit{dom}(\sigma_0) = \{x \mid (X \vdash^? x) \in \mathsf{D}, X \in \mathcal{X}^2_0\}$ and for any $x \in \mathit{dom}(\sigma_0)$, $x\sigma_0 = X\Sigma\!\downarrow$. Note that when $X \in \mathcal{X}^2_0$, $X\Sigma \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}})$ since $\mathcal{A}\mathcal{X}_0 = \emptyset$.
- for $1 \leq k \leq n$ we define $\mathit{dom}(\sigma_k) = \{x \mid (X \vdash^? x) \in \mathsf{D}, X \in \mathcal{X}^2_k\}$ and for any $x \in \mathit{dom}(\sigma_k)$, $x\sigma_k = (X\Sigma)(\Phi\sigma_{k-1})\!\downarrow$.

We also note that the empty constraint system $\mathcal{C}_\emptyset$ is satisfiable as $\mathit{Sol}(\mathcal{C}_\emptyset) = \{(\emptyset, \emptyset)\}$.

Sometimes we need to restrict the set of solutions. Given a predicate π on second-order solutions we write $\mathit{Sol}^\pi(C)$ for the solutions of $\mathcal{C}$ that satisfy π, i.e. $\mathit{Sol}^\pi(C) = \{(\Sigma, \sigma) \mid (\Sigma, \sigma) \in \mathit{Sol}(C) \text{ and } \pi(\Sigma)\}$. The predicate which holds on all second-order solutions is denoted by $\top$ and we have that $\mathit{Sol}^\top(C) = \mathit{Sol}(C)$.

6.2 Most general solutions

Similarly to the notion of most general unifiers of two terms, we define the *most general solutions* of a constraint system that describe the shape of the all solutions of a constraint system. Unlike syntactic most general unifiers, the most general solutions of a constraint system may not be unique (even up to variable renaming) and may also require to *create* new variables.

Definition 14. *Let $\mathcal{C}$ be a constraint system and π a predicate on second-order solutions. We define the* most general solutions of $\mathcal{C}$ that satisfy π, *denoted $\mathit{mgs}^\pi(\mathcal{C})$, as a set of substitutions such that:*

- *for all $(\Sigma, \sigma) \in \mathit{Sol}^\pi(\mathcal{C})$, there exist $\Sigma_0 \in \mathit{mgs}^\pi(\mathcal{C})$ and a substitution Σ_1 such that $\Sigma = \Sigma_0\Sigma_1|_{\mathit{vars}^2(\mathcal{C})}$.*
- *for all $\Sigma_0 \in \mathit{mgs}^\pi(\mathcal{C})$, $\mathit{dom}(\Sigma_0) \subseteq \mathit{vars}^2(\mathcal{C})$ and for all fresh bijective renaming Σ_1 from $\mathit{vars}^2(\Sigma_0) \cup \mathit{vars}^2(\mathcal{C}) \setminus \mathit{dom}(\Sigma_0)$ to $\mathcal{N}_{\mathsf{pub}}$, there exists σ such that $(\Sigma_0\Sigma_1|_{\mathit{vars}^2(\mathcal{C})}, \sigma) \in \mathit{Sol}^\pi(\mathcal{C})$*

We suppose that all substitutions in $\mathit{mgs}^\pi(\mathcal{C})$ are distinct modulo renaming of variables.

The first condition is very similar to the most general unifier as it specifies that any solution of a constraint system is an instance of one of the most general solutions. However, this is not sufficient as for instance the identity substitution would satisfy it. Therefore, the second condition captures that each most general solution is a non trivial solution: replacing all second-order variables in the most general solution by distinct, fresh names yields a valid solution.

6.3 Symbolic processes and symbolic semantics

A symbolic process is a pair $(\mathcal{P}, \mathcal{C})$ where $\mathcal{P}$ is a multiset of plain processes and $\mathcal{C}$ is a constraint system. We define the symbolic semantics through a labelled transition relation $A \xrightarrow{a}_{\mathsf{s}} B$ where A, B are symbolic processes and a is an *action*. The alphabet of actions $\mathcal{A}_{\mathsf{s}}$ consists of

- the set of input actions $X(Y)$ where X and Y are recipe variables;

- the set of output actions $\overline{X}\langle\mathsf{ax}\rangle$ where X is a recipe variable and ax is an axiom.

In a transition $A \xrightarrow{a}_{\mathsf{s}} B$, a is either the empty word ε or a letter from $\mathcal{A}_{\mathsf{s}}$. The transition relation is defined by the rules given in fig. 7. As a notational shortcut, we interpret a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ as the formula $x_1 =^? t_1 \wedge \ldots \wedge x_n =^? t_n$ and also use the notation $syn_{\mathcal{R}}(u \neq^? v)$ (in rule (s-Else)) defined in lemma 6. Finally, note that $mgu_{\mathcal{R}}(t =^? t)$ may not be the identity substitution, as it enforces that t must be instantiated to a message (see Example 3).

$$(\mathcal{P} \cup \{\!\{0\}\!\}, \mathcal{C}) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P}, \mathcal{C}) \qquad \text{(s-Null)}$$

$$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q_1\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge \sigma)) \qquad \text{(s-Then)}$$
$$\text{if } \sigma \in mgu_{\mathcal{R}}(u\mu\!\downarrow =^? v\mu\!\downarrow)$$

$$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q_2\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge \varphi)) \qquad \text{(s-Else)}$$
$$\text{where } \varphi = syn_{\mathcal{R}}(u\mu\!\downarrow \neq^? v\mu\!\downarrow)$$

$$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.Q_1, v(x).Q_2\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q_1, Q_2\{x \to t\}\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge \sigma)) \qquad \text{(s-Comm)}$$
$$\text{if } \sigma \in mgu_{\mathcal{R}}(u\mu\!\downarrow =^? v\mu\!\downarrow, t\mu\!\downarrow =^? t\mu\!\downarrow)$$

$$(\mathcal{P} \cup \{\!\{P \mid Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{P, Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \qquad \text{(s-Par)}$$

$$(\mathcal{P} \cup \{\!\{u(x).Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{Y(X)}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi, \mathsf{D} \wedge X \vdash^? x \wedge Y \vdash^? y, \mathsf{E}^1 \wedge \sigma)) \qquad \text{(s-In)}$$
$$\text{if } \sigma \in mgu_{\mathcal{R}}(y =^? u\mu\!\downarrow) \text{ and } X, Y \in \mathcal{X}^2_{:|\Phi|}$$

$$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1)) \xrightarrow{\overline{Y}\langle \mathsf{ax}_n\rangle}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi \cup \{\mathsf{ax}_n \mapsto t\sigma\!\downarrow\}, \mathsf{D} \wedge Y \vdash^? y, \mathsf{E}^1 \wedge \sigma)) \qquad \text{(s-Out)}$$
$$\text{if } \sigma \in mgu_{\mathcal{R}}(y =^? u\mu\!\downarrow, t\mu\!\downarrow =^? t\mu\!\downarrow),\ y \text{ is fresh, } X \in \mathcal{X}^2_{:n} \text{ and } n = |\Phi| + 1$$

where u, v and t are terms, $x \in \mathcal{X}^1$, $X, Y \in \mathcal{X}^2$, $\mathsf{ax}_n \in \mathcal{AX}$ and $\mu = mgu(\mathsf{E}^1|_=)$.

Figure 7: Symbolic semantics

Given a word $w \in \mathcal{A}_{\mathsf{s}}^*$, we define the relation $A \overset{w}{\Rightarrow}_{\mathsf{s}} B$ as $A \xrightarrow{a_1}_{\mathsf{s}} \ldots \xrightarrow{a_n}_{\mathsf{s}} B$ where $w = a_1 \ldots a_n$, $n \in \mathbb{N}$ and all second order variables in w are pairwise distinct. An example of symbolic execution tree of processes introduced in example 5 is provided in fig. 8.

We now show that our symbolic semantics is sound and complete.

Lemma 7. *Let $(\mathcal{P}, \mathcal{C})$ be a symbolic process. We have that*

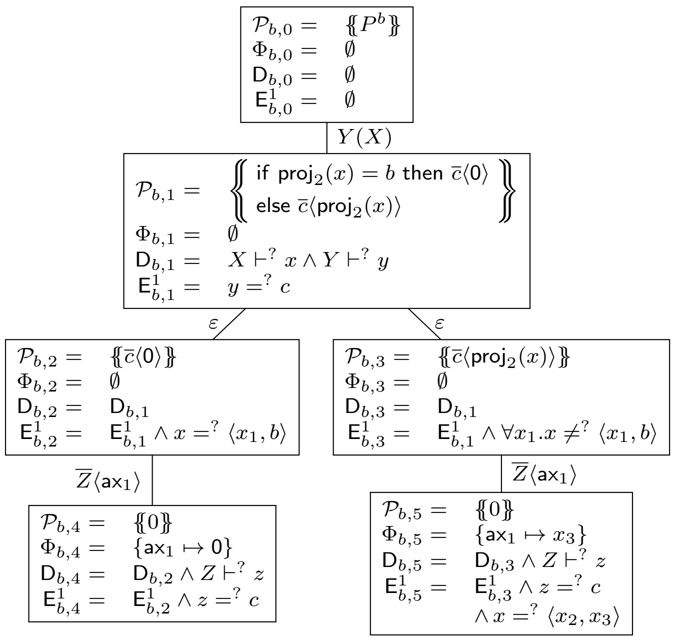
- *if $(\mathcal{P}, \mathcal{C}) \overset{\mathsf{tr}_s}{\Longrightarrow}_{\mathsf{s}} (\mathcal{Q}, \mathcal{C}')$ and $(\Sigma, \sigma) \in Sol(\mathcal{C}')$ then $(\mathcal{P}\sigma, \Phi(\mathcal{C})\sigma\!\downarrow) \overset{\mathsf{tr}_s\Sigma}{\Longrightarrow}_{\mathsf{c}} (\mathcal{Q}\sigma, \Phi(\mathcal{C}')\sigma\!\downarrow)$;*
- *if $(\Sigma, \sigma) \in Sol(\mathcal{C})$ and $(\mathcal{P}\sigma, \Phi(\mathcal{C})\sigma) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (Q, \Phi)$ then $(\mathcal{P}, \mathcal{C}) \overset{\mathsf{tr}_s}{\Longrightarrow}_{\mathsf{s}} (\mathcal{Q}', \mathcal{C}')$ and $(\Sigma', \sigma') \in Sol(\mathcal{C}')$ such that $\Sigma' \succeq \Sigma$, $\mathcal{Q} = \mathcal{Q}'\sigma'$, $\mathsf{tr} = \mathsf{tr}_s\Sigma'$ and $\Phi = \Phi(\mathcal{C}')\sigma'\!\downarrow$.*

Note that taking a ground plain process P and $\mathcal{C} = \mathcal{C}_\emptyset$ the second item simplifies to

- if $(\{\!\{P\}\!\}, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{Q}, \Phi)$ then $(\{\!\{P\}\!\}, \mathcal{C}_\emptyset) \overset{\mathsf{tr}_s}{\Longrightarrow}_{\mathsf{s}} (\mathcal{Q}', \mathcal{C}')$ and $(\Sigma', \sigma') \in Sol(\mathcal{C}')$ such that $\mathcal{Q} = \mathcal{Q}'\sigma'$, $\mathsf{tr} = \mathsf{tr}_s\Sigma'$ and $\Phi = \Phi(\mathcal{C}')\sigma'\!\downarrow$.

6.4 Partition Tree

To decide trace and observational equivalence, we introduce the notion of *partition tree* of two plain processes, say P_1 and P_2. The aim is to build the (finite) tree of all possible symbolic executions (regrouping processes reached by ε transitions in a same node), and additionally partition solutions of constraint systems such all concrete processes represented by a node



$\mathcal{P}_{b,0} = \{\!\{P^b\}\!\}$
$\Phi_{b,0} = \emptyset$
$\mathsf{D}_{b,0} = \emptyset$
$\mathsf{E}^1_{b,0} = \emptyset$
$Y(X)$
$\mathcal{P}_{b,1} = \{\!\{$ if $\mathsf{proj}_2(x) = b$ then $\overline{c}\langle 0\rangle$ else $\overline{c}\langle \mathsf{proj}_2(x)\rangle \}\!\}$
$\Phi_{b,1} = \emptyset$
$\mathsf{D}_{b,1} = X \vdash^? x \wedge Y \vdash^? y$
$\mathsf{E}^1_{b,1} = y =^? c$
ε
ε
$\mathcal{P}_{b,2} = \{\!\{\overline{c}\langle 0\rangle\}\!\}$
$\Phi_{b,2} = \emptyset$
$\mathsf{D}_{b,2} = \mathsf{D}_{b,1}$
$\mathsf{E}^1_{b,2} = \mathsf{E}^1_{b,1} \wedge x =^? \langle x_1, b\rangle$
$\mathcal{P}_{b,3} = \{\!\{\overline{c}\langle \mathsf{proj}_2(x)\rangle\}\!\}$
$\Phi_{b,3} = \emptyset$
$\mathsf{D}_{b,3} = \mathsf{D}_{b,1}$
$\mathsf{E}^1_{b,3} = \mathsf{E}^1_{b,1} \wedge \forall x_1 . x \neq^? \langle x_1, b\rangle$
$\overline{Z}\langle \mathsf{ax}_1\rangle$
$\overline{Z}\langle \mathsf{ax}_1\rangle$
$\mathcal{P}_{b,4} = \{\!\{0\}\!\}$
$\Phi_{b,4} = \{\mathsf{ax}_1 \mapsto 0\}$
$\mathsf{D}_{b,4} = \mathsf{D}_{b,2} \wedge Z \vdash^? z$
$\mathsf{E}^1_{b,4} = \mathsf{E}^1_{b,2} \wedge z =^? c$
$\mathcal{P}_{b,5} = \{\!\{0\}\!\}$
$\Phi_{b,5} = \{\mathsf{ax}_1 \mapsto x_3\}$
$\mathsf{D}_{b,5} = \mathsf{D}_{b,3} \wedge Z \vdash^? z$
$\mathsf{E}^1_{b,5} = \mathsf{E}^1_{b,3} \wedge z =^? c \wedge x =^? \langle x_2, x_3\rangle$

Figure 8: Symbolic execution tree of P^b, $b \in \{0, 1\}$

(which contains a set of symbolic processes) are statically equivalent. This partition tree will allow us to construct a witness violating trace equivalence, resp. labelled bisimilarity, when $P_1 \not\approx_t P_2$, resp. $P_1 \not\approx_\ell P_2$. The partition tree will be a finite tree whose nodes are labelled by *configurations.*

Definition 15. *A* configuration *is a triple* (Γ, π, ℓ) *where*

- Γ *is a set of symbolic processes,*
- π *is a predicate on second-order solutions such that if* $\pi(\Sigma)$ *and* $\Sigma' \succeq \Sigma$ *then* $\pi(\Sigma')$*, and*
- $\ell \in \mathcal{A}_{\mathsf{s}} \cup \{\varepsilon\}$

such that

1. *for all* $(\mathcal{P}, \mathcal{C}) \in \Gamma$*,* $Sol^\pi(\mathcal{C}) \neq \emptyset$ *and* $|mgs^\pi(\mathcal{C})| = 1$*;*
2. *if* $(\mathcal{P}_1, \mathcal{C}_1), (\mathcal{P}_2, \mathcal{C}_2) \in \Gamma$*,* $(\Sigma, \sigma_1) \in Sol^\pi(\mathcal{C}_1)$ *then* $(\Sigma, \sigma_2) \in Sol^\pi(\mathcal{C}_2)$ *and* $\Phi(\mathcal{C}_1)\sigma_1 \sim \Phi(\mathcal{C}_2)\sigma_2$*.*

When a node n of a partition tree is labelled by (Γ, π, ℓ) we denote by $\Gamma(n)$, $\pi(n)$, and $\ell(n)$ the corresponding elements of the configuration. Moreover, it follows from the above definition that all constraint systems in $\Gamma(n)$ for a given node n have a same, unique most general solution, which we denote $mgs(n)$.

Definition 16. *Let* P_1 *and* P_2 *be two closed plain processes. A* partition tree T of P_1 and P_2 *is a finite tree whose nodes are labelled by configurations. It also verifies, for all nodes* n *and* $(\mathcal{P}, \mathcal{C}) \in \Gamma(n)$*:*

1. *the initial processes are in the root:*

   if n is the root of the tree then $\ell(n) = \varepsilon$, $\pi(n) = \top$ and $\Gamma(n)$ contains $(\{\!\!\{P_1\}\!\!\}, \mathcal{C}_\emptyset)$ and $(\{\!\!\{P_2\}\!\!\}, \mathcal{C}_\emptyset)$;

2. *nodes are closed under ε-transition:*

   if $(\mathcal{P}, \mathcal{C}) \overset{\varepsilon}{\Rightarrow}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$ and $Sol^{\pi(n)}(\mathcal{C}') \neq \emptyset$ then $(\mathcal{P}', \mathcal{C}') \in \Gamma$;

3. *completeness of the partition tree:*

   if $(\mathcal{P}, \mathcal{C}) \overset{\ell}{\Rightarrow}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$ and $(\Sigma, \sigma) \in Sol^{\pi(n)}(\mathcal{C}')$ then there exists n' child of n s.t. $(\mathcal{P}', \mathcal{C}') \in \Gamma(n')$, $\ell(n') = \ell$ and $(\Sigma', \sigma) \in Sol^{\pi(n')}(\mathcal{C}')$ for some Σ';

Besides, if n_c is a child node of n and $(\mathcal{P}_c, \mathcal{C}_c) \in \Gamma(n_c)$:

4. *predicates are refined along branches: $\pi(n_c) \subseteq \pi(n)$;*

5. *soundness of the partition tree:*

   if $(\Sigma, \sigma) \in Sol^{\pi(n)}(\mathcal{C})$, $(\Sigma_c, \sigma_c) \in Sol^{\pi(n_c)}(\mathcal{C}_c)$ and $\Sigma_c \succeq \Sigma$, then $\Gamma(n_c)$ contains all $(\mathcal{P}', \mathcal{C}')$ such that $(\mathcal{P}, \mathcal{C}) \xRightarrow{\ell(n_c)}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$ and $\Phi(\mathcal{C}_c)\sigma_c \sim \Phi(\mathcal{C}')\sigma'$ for some σ' such that $(\Sigma_c, \sigma') \in Sol(\mathcal{C}')$.

Finally, we also impose a property of homogeneity of the mgs' of the nodes:

6. *for all subset S of nodes*

   if $E = \{X =^? X mgs(n) \mid X \in vars^2(T),\ n \in S\}$ and $\Upsilon = mgu(E) \neq \bot$

   then for all $n \in S$, $mgs(n\Upsilon) \neq \emptyset$ entails $mgs(n\Upsilon) = \{mgs(n)\Upsilon|_{vars^2(\mathcal{C})}\}$.

We denote by $\mathsf{PTree}(P_1, P_2)$ the (infinite) set of all partition trees of P_1 and P_2.

Example 8. Two partition trees are presented in fig. 9. They use notations of fig. 8, $\mathcal{E}_i^b = (\{\!\!\{\mathcal{P}_{b,i}\}\!\!\}, (\Phi_{b,i}, \mathsf{D}_{b,i}, \mathsf{E}^1_{b,i}))$, $\mathcal{E}_Q^b = (\overline{c}\langle \mathsf{proj}_2(x)\rangle, \mathcal{C}_{b,1})$ and

$$\mathcal{E}_Q^f = (\{\!\!\{0\}\!\!\}, (\Phi_f, \mathsf{D}_f, \mathsf{E}^1_f)) \qquad \mathsf{D}_f = \{X \vdash^? x, Y \vdash^? y, Z \vdash^? z\}$$
$$\Phi_f = \{\mathsf{ax}_1 \mapsto x_2\} \qquad \mathsf{E}^1_f = \{y =^? c, z =^? c, x = \langle x_1, x_2\rangle\}$$

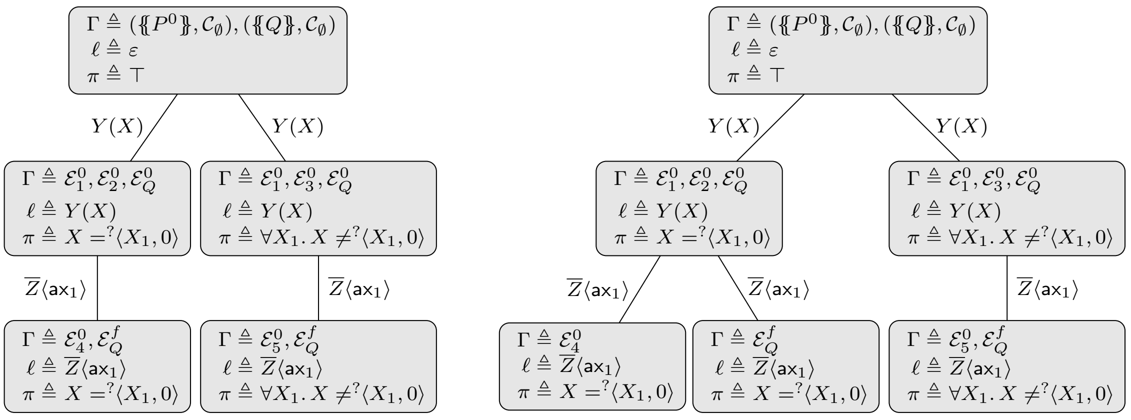
In this example, second-order predicates π are described by second-order formulas φ^2 (casting the satisfiability relation of our first-order logic in the natural way).

Given $T \in \mathsf{PTree}(P_1, P_2)$ we write $(\mathcal{P}, \mathcal{C}), n \xrightarrow{\ell'}_T (\mathcal{P}', \mathcal{C}'), n'$ when

- n is a node in T with $(\mathcal{P}, \mathcal{C}) \in \Gamma(n)$, and
- n has a child node n' with $(\mathcal{P}', \mathcal{C}') \in \Gamma(n')$, and
- $(\mathcal{P}, \mathcal{C}) \xrightarrow{\ell'}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$.

As for the other semantics, we write $(\mathcal{P}, \mathcal{C}), n \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}', \mathcal{C}'), n'$ when $(\mathcal{P}, \mathcal{C}), n \xrightarrow{\ell_1}_T \ldots \xrightarrow{\ell_k}_T (\mathcal{P}', \mathcal{C}'), n'$ with $\mathsf{tr} = \ell_1 \ldots \ell_k$, $k \in \mathbb{N}$. Note that when $\mathsf{tr} = \varepsilon$, we write $(\mathcal{P}, \mathcal{C}), n \overset{\varepsilon}{\Rightarrow}_T (\mathcal{P}', \mathcal{C}'), n$ when $(\mathcal{P}', \mathcal{C}') \in \Gamma$ and $(\mathcal{P}, \mathcal{C}) \overset{\varepsilon}{\Rightarrow}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$.

As previously mentioned, we will use the partition tree to generate a witness that violates equivalence. To obtain a complexity upper bound, we have to bound the size of these



$\Gamma \triangleq (\{\!\!\{P^0\}\!\!\}, \mathcal{C}_\emptyset), (\{\!\!\{Q\}\!\!\}, \mathcal{C}_\emptyset)$
$\ell \triangleq \varepsilon$
$\pi \triangleq \top$
$Y(X)$
$\Gamma \triangleq \mathcal{E}_1^0, \mathcal{E}_2^0, \mathcal{E}_Q^0$
$\ell \triangleq Y(X)$
$\pi \triangleq X =^? \langle X_1, 0\rangle$
$\Gamma \triangleq \mathcal{E}_1^0, \mathcal{E}_3^0, \mathcal{E}_Q^0$
$\ell \triangleq Y(X)$
$\pi \triangleq \forall X_1. X \neq^? \langle X_1, 0\rangle$
$\overline{Z}\langle \mathsf{ax}_1\rangle$
$\Gamma \triangleq \mathcal{E}_4^0, \mathcal{E}_Q^f$
$\Gamma \triangleq \mathcal{E}_5^0, \mathcal{E}_Q^f$
$\Gamma \triangleq \mathcal{E}_4^0$
$\Gamma \triangleq \mathcal{E}_Q^f$
$\ell \triangleq \overline{Z}\langle \mathsf{ax}_1\rangle$

(a) Partition tree $T_0 \in \mathsf{PTree}(P^0, Q)$ (b) Partition tree $T_1 \in \mathsf{PTree}(P^1, Q)$

Figure 9: Example of partition trees

witnesses. We show that for convergent subterm destructor rewriting system, we can always obtain such a witness whose size (in DAG form) is exponential in the size (in DAG form) of the plain processes and the rewriting system.

Theorem 5. *There exists $p \in \mathbb{N}$ such that for all convergent subterm destructor rewriting system $\mathcal{R}$, closed plain processes P and Q there exists a partition tree $T \in \mathsf{PTree}(P, Q)$ such that for all node n in T, for all $\Sigma \in mgs(n)$, $|\Sigma|_{\mathsf{dag}} < 2^{(|P|_{\mathsf{dag}}+|Q|_{\mathsf{dag}}+|\mathcal{R}|_{\mathsf{dag}})^p}$.*

In part III, we prove this theorem by presenting a procedure that solves $\texttt{PartitionTree}_p$ for a constant p. (The constant p is derived from the measure used for proving termination of the procedure.)

7 Decision procedures

7.1 Pure π-calculus

Static equivalence In the pure π-calculus, two frames Φ and Φ' of same domain can only contain (private or public) names. Hence, since $\mathcal{F} = \emptyset$, static equivalence can be characterized as follows:

$$\Phi \sim \Phi' \text{ if and only if } dom(\Phi) = dom(\Phi') \wedge \bigwedge_{\xi,\xi' \in T} (\xi\Phi = \xi'\Phi \Leftrightarrow \xi\Phi' = \xi'\Phi') \qquad (5)$$

where $T = dom(\Phi) \cup (names(\Phi, \Phi') \cap \mathcal{N}_{\mathsf{pub}})$.

The characterization provided by eq. (5) is easily implementable by a quadratic procedure storing a finite number of counters only, each of which being bounded by the size of T which is linear in the size of the problem and whose representation in binary is of logarithmic size. In particular, note that there is no need for T to be computed and stored (since it can be read on-the-fly directly on the input) and analogously, testing an equality $\xi\Phi = \xi'\Phi$ does not require additional storage space. Therefore, we obtain the following upper bound.

Theorem 6. $\mathsf{Equiv}_{\emptyset,\emptyset}^{\sim}$ *is* LOGSPACE.

Trace equivalence and labelled bisimilarity Given two closed extended processes A_0 and A_1, let us recall the following characterization of trace equivalence:

$$A_0 \approx_t A_1 \text{ if and only if } \forall i \in \{0,1\}, \forall A_i \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A'_i. \exists A_{1-i} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A'_{1-i}. \Phi(A'_i) \sim \Phi(A'_{1-i}) \qquad (6)$$

Without non-constant function symbols, there is a polynomial number of transitions from a process A (seeing all names in $\mathcal{N}_{\mathsf{pub}} - names(A)$ as a single "fresh name" entity). In particular, guessing a reduction $A \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A'$ from A is feasible in polynomial time (by a non-deterministic Turing machine). This provides a polynomial-time decision procedure using an alternating Turing machine, starting from a universal state and with at most one quantifier alternation (which is a typical characterization of the complexity class Π_2).

Theorem 7. $\mathsf{Equiv}^{\approx_t}_{\emptyset,\emptyset}$ *is* Π_2*.*

On the other hand, labelled bisimilarity may be characterized as:

$$A_0 \approx_\ell A_1 \text{ if and only if } \begin{cases} \Phi(A_0) \sim \Phi(A_1) \\ \forall i \in \{0,1\},\ \forall A_i \xrightarrow{w}_{\mathsf{c}} A'_i,\ \exists A_{1-i} \overset{w}{\Rightarrow}_{\mathsf{c}} A'_{1-i},\ A'_i \approx_\ell A'_{1-i} \end{cases} \qquad (7)$$

and similarly to above, this provides a straightforward procedure in alternating polynomial time (which is a classical characterization of PSPACE).

Theorem 8. $\mathsf{Equiv}^{\approx_\ell}_{\emptyset,\emptyset}$ *is* PSPACE*.*

7.2 Applied π-calculus

Static equivalence As discussed previously existing procedures for deciding static equivalence for subterm convergent rewrite systems [AC06, CDK12, CBC10] are actually exponential in the signature or equational theory, i.e., they show that for any signature $\mathcal{F}$ and subterm convergent rewriting system $\mathcal{R}$ the problem $\mathsf{Equiv}^{\sim}_{\mathcal{R},\mathcal{F}}$ is in PTIME. As many modern tools allow the user to define the signature and rewrite system, it is often more interesting to study the complexity of $\mathsf{Equiv}^{\sim}$.

Theorem 9. $\mathsf{Equiv}^{\sim}$ *for convergent subterm destructor rewriting systems is in co*NP*.*

Trace equivalence In this section we restrict ourselves to a convergent subterm destructor rewriting system $\mathcal{R}$.

Lemma 8. *Let T be a partition tree, n, n' nodes of T and $(\mathcal{P}_1, \mathcal{C}_1), (\mathcal{P}_2, \mathcal{C}_2) \in \Gamma(n)$. If*

(H_1) $(\mathcal{P}_1, \mathcal{C}_1), n \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}'_1, \mathcal{C}'_1), n'$

(H_2) $(\mathcal{P}_2\sigma_2, \Phi(\mathcal{C}_2)\sigma_2\!\downarrow) \overset{\mathsf{tr}\Sigma'}{\Longrightarrow}_{\mathsf{c}} (\mathcal{P}, \Phi)$

(H_3) $(\Sigma', \sigma_1) \in Sol^{\pi(n')}(\mathcal{C}'_1)$, $(\Sigma, \sigma_2) \in Sol(\mathcal{C}_2)$

(H_4) $\Sigma' \succeq \Sigma$

(H_5) $\Phi(\mathcal{C}'_1)\sigma_1 \sim \Phi$

then $(\mathcal{P}_2, \mathcal{C}_2), n \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}'_2, \mathcal{C}'_2), n'$ *for some* $(\mathcal{P}'_2, \mathcal{C}'_2)$*.*

Proof. We prove the result by induction on $|\mathsf{tr}|$.
case $\mathsf{tr} = \varepsilon$*:* The result directly follows from $n = n'$ and $(P_2, \mathcal{C}_2), n \overset{\varepsilon}{\Rightarrow}_T (P_2, \mathcal{C}_2), n$.

case $\mathsf{tr} = \ell \cdot \mathsf{tr}'$*:* First of all, let us write in (H_1) for some $(\tilde{\mathcal{P}}_1, \tilde{C}_1)$ and node $\tilde{n}$ of T:

$$(\mathcal{P}_1, \mathcal{C}_1), n \overset{\ell}{\Rightarrow}_T (\tilde{\mathcal{P}}_1, \tilde{\mathcal{C}}_1), \tilde{n} \overset{\mathsf{tr}'}{\Rightarrow}_T (\mathcal{P}_1', \mathcal{C}_1'), n'$$

In particular note that, by item 4 of definition 16 and since Σ' satisfies $\pi(n')$ by (H_3), Σ' also satisfies $\pi(\tilde{n})$. Besides, we have in (H_2) for some $(\tilde{\mathcal{P}}, \tilde{\Phi},)$:

$$(\mathcal{P}_2\sigma_2, \Phi(\mathcal{C}_2)\sigma_2{\downarrow}) \xRightarrow{\ell\Sigma'}_{\mathsf{c}} (\tilde{\mathcal{P}}, \tilde{\Phi}) \xRightarrow{\mathsf{tr}\Sigma'}_{\mathsf{c}} (\mathcal{P}, \Phi)$$

Using (H_3) and completeness of the symbolic semantics (second item of lemma 7), we can write the intermediary process under the form $(\tilde{\mathcal{P}}, \tilde{\Phi}) = (\tilde{\mathcal{P}}_2\tilde{\sigma}_2, \Phi(\tilde{\mathcal{C}}_2)\tilde{\sigma}_2)$ where

$$(\mathcal{P}_2, \mathcal{C}_2) \overset{\ell}{\Rightarrow}_{\mathsf{c}} (\tilde{\mathcal{P}}_2, \tilde{\mathcal{C}}_2) \qquad\qquad (\tilde{\Sigma}, \tilde{\sigma}_2) \in \mathit{Sol}(\tilde{\mathcal{C}}_2) \qquad\qquad \tilde{\Sigma} \succeq \Sigma'$$

Observe that $\tilde{\Sigma}$ satisfies $\pi(\tilde{n})$ just like Σ', since $\pi(\tilde{n})$ is stable by domain extention by definition. Observe also that, by (H_5), $\Phi(\tilde{\mathcal{C}}_2)\tilde{\sigma}_2 = \tilde{\Phi} \sim \Phi(\tilde{\mathcal{C}}_1\sigma_1)$. In particular, by item 5 of definition 16, we know that $\Gamma(\tilde{n})$ contains $(\tilde{P}_2, \tilde{\mathcal{C}}_2)$. Hence $(\mathcal{P}_2, \mathcal{C}_2), n \overset{\ell}{\Rightarrow}_T (\tilde{\mathcal{P}}_2, \tilde{\mathcal{C}}_2), \tilde{n}$ and the conclusion follows from induction hypothesis. ☐

Lemma 9. *Let P_1, P_2 be two closed plain processes, $T \in \mathsf{PTree}(P_1, P_2)$ and n_0 the root of T.*

$$\begin{gathered}(\{\!\{P_1\}\!\}, \emptyset) \sqsubseteq_t (\{\!\{P_2\}\!\}, \emptyset)\\ \textit{iff}\\ \textit{if } (\{\!\{P_1\}\!\}, \mathcal{C}_\emptyset), n_0 \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}_A, \mathcal{C}_A), n \textit{ then } (\{\!\{P_2\}\!\}, \mathcal{C}_\emptyset), n_0 \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}_B, \mathcal{C}_B), n\end{gathered}$$

Proof. Suppose that n is labelled by (Γ, π, ℓ). We prove both directions separately.

($\Rightarrow$) We have, by definition of a partition tree (definition 16, item 1) that $(\{\!\{P_1\}\!\}, \mathcal{C}_\emptyset)$ and $(\{\!\{P_2\}\!\}, \mathcal{C}_\emptyset)$ are both in the root node of T. Moreover, by Definition 15, there exists $(\Sigma, \sigma) \in \mathit{Sol}^\pi(\mathcal{C}_A)$ and by the soundness of the symbolic semantics (Lemma 7), we have that $(\{\!\{P_1\}\!\}, \emptyset) \xRightarrow{\mathsf{tr}\Sigma}_{\mathsf{c}} (\mathcal{P}_A\sigma, \Phi(\mathcal{C}_A)\sigma{\downarrow})$ and $(\emptyset, \emptyset) \in \mathit{Sol}(\mathcal{C}_\emptyset)$. Suppose, by contradiction that for all $(\mathcal{P}_B, \mathcal{C}_B)$, $(\{\!\{P_2\}\!\}, \mathcal{C}_\emptyset), n_0 \overset{\mathsf{tr}}{\not\Rightarrow}_T (\mathcal{P}_B, \mathcal{C}_B), n$. By lemma 8, for all $(\mathcal{P}, \Phi)$, either $(\{\!\{P_2\}\!\}, \emptyset) \overset{\mathsf{tr}\Sigma}{\not\Longrightarrow}_{\mathsf{c}} (\mathcal{P}, \Phi)$ or $(\{\!\{P_2\}\!\}, \emptyset) \xRightarrow{\mathsf{tr}\Sigma}_{\mathsf{c}} (\mathcal{P}, \Phi)$ and $\Phi(\mathcal{C}_A)\sigma{\downarrow} \not\sim \Phi$ which implies $(\{\!\{P_1\}\!\}, \emptyset) \not\sqsubseteq_t (\{\!\{P_2\}\!\}, \emptyset)$ leading to a contradiction.

($\Leftarrow$) Suppose $(\{\!\{P_1\}\!\}, \emptyset) \xRightarrow{\mathsf{tr}_c}_{\mathsf{c}} (\mathcal{P}, \Phi)$. We need to show that there exists $(\mathcal{P}', \Phi')$ such that $(\{\!\{P_2\}\!\}, \emptyset) \xRightarrow{\mathsf{tr}_c}_{\mathsf{c}} (\mathcal{P}', \Phi')$ and $\Phi \sim \Phi'$. By a simple induction on $|\mathsf{tr}_c|$ we show that there exist a word tr, a node n labelled (Γ, π, ℓ), a symbolic process $(\mathcal{P}_A, \mathcal{C}_A) \in \Gamma$ and $(\Sigma, \sigma_A) \in \mathit{Sol}(\mathcal{C}_A)$ such that $(\{\!\{P_1\}\!\}, \mathcal{C}_\emptyset), n_0 \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}_A, \mathcal{C}_A), n$, $\Phi(\mathcal{C}_A)\sigma_A{\downarrow} = \Phi$ and $\mathsf{tr}_c\Phi{\downarrow} = \mathsf{tr}\Sigma\Phi{\downarrow}$: the proof is done by application of lemma 7 (completeness of the symbolic semantics) and item 2 for the base case, respectively item 3 for the inductive case, of definition 16. By hypothesis, there exists $(\mathcal{P}_B, \mathcal{C}_B)$ such that $(\{\!\{P_2\}\!\}, \mathcal{C}_\emptyset), n_0 \overset{\mathsf{tr}}{\Rightarrow}_T (\mathcal{P}_B, \mathcal{C}_B), n$. Hence $(\mathcal{P}_B, \mathcal{C}_B) \in \Gamma$ and so by definition 15, there exists σ_B such that $(\Sigma, \sigma_B) \in Sol(\mathcal{C}_B)$ and $\Phi = \Phi(\mathcal{C}_A)\sigma_A\!\downarrow \sim \Phi(\mathcal{C}_B)\sigma_B\!\downarrow$. By Lemma 7, $(\{\!\{P_2\}\!\}, \emptyset) \overset{\mathsf{tr}\Sigma}{\Longrightarrow}_{\mathsf{c}} (\mathcal{P}_B\sigma_B, \Phi(\mathcal{C}_B)\sigma_B\!\downarrow)$. As $\mathsf{tr}_c\Phi\!\downarrow = \mathsf{tr}\Sigma\Phi\!\downarrow$ and $\Phi \sim \Phi(\mathcal{C}_B)\sigma_B\!\downarrow$ we have that $\mathsf{tr}_c\Phi(\mathcal{C}_B)\sigma_B\!\downarrow = \mathsf{tr}\Sigma\Phi(\mathcal{C}_B)\sigma_B\!\downarrow$. Hence, we conclude that $(\{\!\{P_2\}\!\}, \emptyset) \overset{\mathsf{tr}_c}{\Longrightarrow}_{\mathsf{c}} (\mathcal{P}_B\sigma_B, \Phi(\mathcal{C}_B)\sigma_B\!\downarrow)$. □

By Lemma 9 and Theorem 5, we obtain the following corollary:

Corollary 1. *There exists $p \in \mathbb{N}$ s.t. if $(\{\!\{P_1\}\!\}, \emptyset) \not\approx_t (\{\!\{P_2\}\!\}, \emptyset)$ then there exists* tr *s.t.*

- $(\{\!\{P_1\}\!\}, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{P}_A, \Phi_A)$ *and for all $(\mathcal{P}_B, \Phi_B)$ such that $(\{\!\{P_2\}\!\}, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{P}_B, \Phi_B)$ we have that $\Phi_A \not\sim \Phi_B$*
- $|\mathsf{tr}|_{\mathsf{dag}} < 2^{|P_1,P_2,\mathcal{R}|^p_{\mathsf{dag}}}$.

Theorem 10. $\mathsf{Equiv}^{\approx_t}$ *is co*NEXP *for convergent subterm destructor rewriting systems.*

Proof. From the previous corollary we obtain a direct non-determinsistic procedure for checking trace inequivalence of $(\{\!\{P_1\}\!\}, \emptyset)$ and $(\{\!\{P_2\}\!\}, \emptyset)$. First, guess the witness tr of exponential size $2^{|P_1,P_2,\mathcal{R}|^p_{\mathsf{dag}}}$ and a reduction $(\mathcal{P}_1, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{P}_A, \Phi_A)$. For each $(\mathcal{P}_B, \Phi_B)$ (at most exponentially many) such that $(\{\!\{P_2\}\!\}, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{P}_B, \Phi_B)$, we verify that $\Phi_A \not\sim \Phi_B$ by guessing a witness of non-equivalence. As $|\mathsf{tr}|_{\mathsf{dag}} < 2^{|P_1,P_2,\mathcal{R}|^p_{\mathsf{dag}}}$ the size of Φ_A and Φ_B is at most exponential: such witnesses on non-static equivalence can therefore be chosen of exponential size by theorem 9. □

Remark 9. We note that our proof of theorem 10 did not rely on item 6 of definition 16: indeed in the case of trace equivalence a weaker definition of a partition tree would have been sufficient. The actual definition will however be necessary when proving the complexity upper bound for labelled bisimilarity.

Labelled bisimilarity As for trace equivalence we restrict ourselves to a convergent subterm destructor rewriting system $\mathcal{R}$. In the case of trace equivalence, a witness that $A \not\approx_t B$ is essentially a sequence of actions tr. In the case of labelled bisimilarity a witness that $A \not\approx_\ell B$ is the tree representing the adversary's strategy. In each node the adversary choses an action and successor process; the children of this node are all possible successors of the other process using the same action.

Definition 17. *A* witness w *is a set of triples (A_0, A_1, ℓ) such that*

1. *A_0 and A_1 are closed extended processes such that $A_0 \sim A_1$ and $\ell \in \mathcal{A} \cup \{\varepsilon\}$;*
2. *there exist $b \in \{0, 1\}$, $\ell \in \mathcal{A} \cup \{\varepsilon\}$ and a transition $A_b \xrightarrow{\ell}_{\mathsf{c}} A'_b$ such that for all reductions $A_{1-b} \overset{\ell}{\Rightarrow}_{\mathsf{c}} A'_{1-b}$,* w *contains a (A'_0, A'_1, ℓ') for some ℓ'.*

Besides, we say that w *is a witness of (A, B) when $(A, B, \ell) \in$* w *for some $\ell \in \mathcal{A} \cup \{\varepsilon\}$. The size of* w *is written* $|\mathsf{w}|$ *(which is not the cardinality of* w *but the sum of the sizes of its elements).*

Lemma 10. *If A_0 and A_1 are two closed extended processes such that $A_0 \sim A_1$, then $A_0 \not\approx_\ell A_1$* **iff** *there exists a witness of (A_0, A_1).*

Proof. First, we observe that $A_0 \not\approx_\ell A_1$ **iff** there exists a binary relation $\mathscr{S}$ on closed extended processes such that $A_0 \mathscr{S} A_1$ and, for all $(B_0, B_1) \in \mathscr{S}$:

1. either $B_0 \not\sim B_1$
2. or there exists $b \in \{0,1\}$, $\ell \in \mathcal{A} \cup \{\varepsilon\}$ and a transition $B_b \xrightarrow{\ell}_{\mathsf{c}} B'_b$ such that $B'_0 \mathscr{S} A'_1$ for all reductions $B_{1-b} \overset{\ell}{\Rightarrow}_{\mathsf{c}} B'_{1-b}$.

Let us call such a relation $\mathscr{S}$ a *simple labelled attack on* (A_0, A_1). Since processes are finite (i.e. $\dot{\rightarrow}_{\mathsf{c}}$ is strongly terminating), $A \not\approx_\ell B$ straightforwardly rephrases to the existence of a simple labelled attack $\mathscr{S}$ such that $A \mathscr{S} B$. Then we prove the expected result by double implication.

$\Rightarrow$ We prove the result by induction on $|A_0, A_1|$. Let $\mathscr{S}$ be a simple labelled attack such that $A_0 \mathscr{S} A_1$. Since $A_0 \sim A_1$, we know that there exists $b \in \{0,1\}$ and $A_b \xrightarrow{\ell}_{\mathsf{c}} A'_b$ such that $A'_0 \mathscr{S} A'_1$ for all $A_{1-b} \overset{\ell}{\Rightarrow}_{\mathsf{c}} A'_{1-b}$. For each such pair (A'_0, A'_1), since $\mathscr{S}$ is also a simple labelled attack on (A'_0, A'_1), we obtain by induction hypothesis a distinguisher $\mathsf{w}_{A'_0, A'_1}$ of (A'_0, A'_1) provided that $A'_0 \sim A'_1$. The conclusion follows by defining

$$\mathsf{w} = \{(A_0, A_1, \ell)\} \cup \bigcup_{\substack{A_{1-b} \overset{\ell}{\Rightarrow}_{\mathsf{c}} A'_{1-b} \\ A'_0 \sim A'_1}} \mathsf{w}_{A'_0, A'_1}$$

which is a witness of (A'_0, A'_1).

$\Leftarrow$ Let w be a witness of (A_0, A_1). By writing $\mathscr{S} = \{(A, B) \mid (A, B, \ell) \in \mathsf{w} \text{ for some } \ell\}$, we easily obtain that $\mathscr{S} \cup \not\sim$ is a labelled attack on (A_0, A_1). □

Example 9. Consider the running examples P^1 and Q introduced in example 5. A witness of $(\{\!\{P^1\}\!\}, \emptyset) \not\approx_\ell (\{\!\{Q\}\!\}, \emptyset)$ is depicted in fig. 10. First, the adversary inputs $\langle 1, 1\rangle$, selecting the transition

$$(Q, \emptyset) \xrightarrow{c(\langle 1,1\rangle)}_{\mathsf{c}} (\{\!\{\overline{c}\langle \mathsf{proj}_2(\langle 1, 1\rangle)\rangle\}\!\}, \emptyset)$$

Whatever the answer of the defender, the adversary can then reach a leave of the tree by choosing the transition

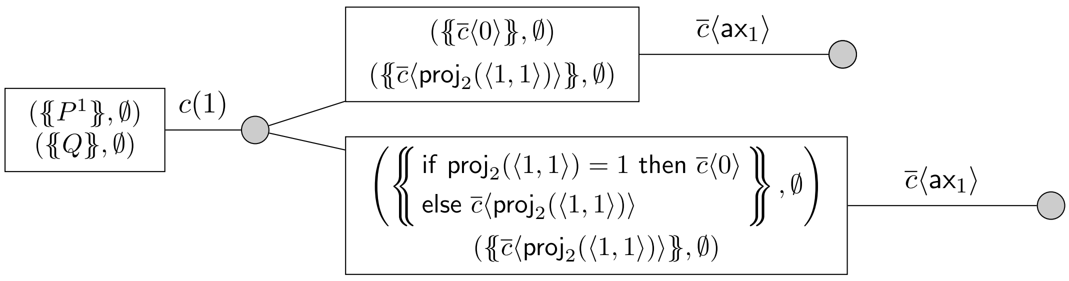
$$(\{\!\{\overline{c}\langle \mathsf{proj}_2(\langle 1, 1\rangle)\rangle\}\!\}, \emptyset) \xrightarrow{\overline{c}\langle \mathsf{ax}_1\rangle}_{\mathsf{c}} (\{\!\{0\}\!\}, \{\mathsf{ax}_1 \mapsto 1\})$$

The defender can indeed not answer to this move without violating static equivalence.

In order to show the existence of a witness whose size is at most exponential we introduce the notion of *symbolic witness*. In essence, a symbolic witness can be seen as a subtree of a partition tree $\mathsf{PTree}(A, B)$ where the labels $\Gamma(n)$ have been restricted to pairs of processes (for internal nodes) or single processes (for leaves, witnessing the end of the bisimulation game).

Definition 18. *Let T be a partition tree, n_0 be a node in T. A* symbolic witness w_s *w.r.t. T is a finite tree whose nodes N verify the following properties:*

1. *N is labelled by a pair (S, n) where n is a node of T and $S \subseteq \Gamma(n)$. We require that S is a singleton if N is a leaf of w_s, and contain exactly two elements otherwise;*



$(\{\!\{P^1\}\!\}, \emptyset)$
$(\{\!\{Q\}\!\}, \emptyset)$
$c(1)$
$(\{\!\{\overline{c}\langle 0\rangle\}\!\}, \emptyset)$
$(\{\!\{\overline{c}\langle \mathsf{proj}_2(\langle 1,1\rangle)\rangle\}\!\}, \emptyset)$
$\overline{c}\langle \mathsf{ax}_1\rangle$
if $\mathsf{proj}_2(\langle 1,1\rangle) = 1$ then $\overline{c}\langle 0\rangle$ else $\overline{c}\langle \mathsf{proj}_2(\langle 1,1\rangle)\rangle$, $\emptyset$
$(\{\!\{\overline{c}\langle \mathsf{proj}_2(\langle 1,1\rangle)\rangle\}\!\}, \emptyset)$
$\overline{c}\langle \mathsf{ax}_1\rangle$

Figure 10: A concrete witness of $(\{\!\{P^1\}\!\}, \emptyset) \not\approx_\ell (\{\!\{Q\}\!\}, \emptyset)$.

2. *if N is labelled $(\{A_0, A_1\}, n)$, it shall exist $b \in \{0,1\}$, a node n' of T and a reduction $A_b, n \xrightarrow{\ell}_T A'_b, n'$ ($\ell \in \mathcal{A}_\mathsf{s} \cup \{\varepsilon\}$) such that:*
   (a) *If $A_{1-b} \overset{\ell}{\Rightarrow}_T A'_{1-b}$ does not hold for any A'_{1-b}, then N has a unique child labelled $(\{A'_b\}, n')$;*
   (b) *otherwise the children of N are the nodes labelled (A'_0, A'_1, n') with $A_{1-b}, n \overset{\ell}{\Rightarrow}_T A'_{1-b}, n'$.*

We say that w_s is a symbolic witness for A, B, n_0 when $root(\mathsf{w}_s)$ is labelled by (A, B, n_0).

We can now define the notion of *solution of a symbolic witness*: a second-order solution of the processes of a node is associated to each node, in a coherent way, i.e., ensuring that solutions of child nodes extend the solution of their parent.

Definition 19. *Let w_s be a symbolic witness w.r.t. a partition tree T. A* solution *of w_s is a function f_sol that maps nodes of w_s to ground second-order substitutions such that*

- *for all nodes N labelled (S, n) and $(\mathcal{P}, \mathcal{C}) \in S$, $(f_\mathsf{sol}(N), \sigma) \in Sol^{\pi(n)}(\mathcal{C})$ for some σ;*
- *for all edges (N_p, N_c) of w_s, $f_\mathsf{sol}(N_c) \succeq f_\mathsf{sol}(N_p)$.*

We denote $Sol(\mathsf{w}_s)$ the set of solutions of w_s.

We can now relate the notion of symbolic witness and the concrete labelled bisimulation.

Lemma 11. *Let T be a partition tree, n a node of T and $A_0, A_1 \in \Gamma(n)$ with $A_i = (\mathcal{P}_i, \mathcal{C}_i)$. Let also $\Sigma, \sigma_0, \sigma_1$ be ground substitutions such that $(\Sigma, \sigma_i) \in Sol^{\pi(n)}(\mathcal{C}_i)$.*

Then, writing $A_i^c = (\mathcal{P}_i\sigma_i, \Phi(\mathcal{C}_i)\sigma_i{\downarrow})$, $A_0^c \not\approx_\ell A_1^c$ **iff** *there exists a symbolic witness w_s of A_0, A_1, n w.r.t. T and $f_\mathsf{sol} \in Sol(\mathsf{w}_s)$ such that $f_\mathsf{sol}(root(\mathsf{w}_s)) = \Sigma$.*

Proof. We prove both directions separately.

$\Rightarrow$ We prove the result by induction on $|A_0^c, A_1^c|$. The conclusion is immediate if $|A_0^c, A_1^c| = 0$ as it yields a contradiction (two empty extended processes are equivalent). Otherwise, since A_0 and A_1 are in the same node n of T, $A_0^c \sim A_1^c$ by definition of a partition tree (definition 16). In particular, under the assumption $A_0^c \not\approx_\ell A_1^c$, we know by lemma 10 that there exists a witness w of (A_0^c, A_1^c). Thus, by definition, there exists $A_b^c \xrightarrow{\ell}_\mathsf{c} A_b'^c = (\mathcal{Q}, \Phi)$ ($b \in \{0,1\}$, $\ell \in \mathcal{A} \cup \{\varepsilon\}$) such that for all reductions $A_{1-b}^c \overset{\ell}{\Rightarrow}_\mathsf{c} A_{1-b}'^c$ such that $A_0'^c \sim A_0'^c$, $(A_0'^c, A_1'^c, \ell') \in \mathsf{w}$ for some ℓ'. In particular for these processes, $A_0'^c \not\approx_\ell A_1'^c$ by lemma 10.

Let us now construct a symbolic witness of A_0, A_1, n. By completeness of the symbolic semantics (lemma 7 second point), we have $\Sigma' \succeq \Sigma$ and $A_b \overset{\ell_s}{\Rightarrow}_{\mathsf{s}} (\mathcal{Q}_s, \mathcal{C})$ such that $(\Sigma', \sigma') \in \mathit{Sol}^{\pi(n)}(\mathcal{C})$ for some σ', $\ell = \ell_s\Sigma'$, $\mathcal{Q} = \mathcal{Q}_s\sigma'$ and $\Phi = \Phi(\mathcal{C})\sigma'\!\downarrow$. Then by completeness of the partition tree (definition 16, item 3), we obtain a tree edge $A_b, n \overset{\ell_s}{\Rightarrow}_T (\mathcal{Q}_s, \mathcal{C}), n'$ for some node n' and $(\Sigma'', \sigma') \in \mathit{Sol}^{\pi(n')}(\mathcal{C})$ for some Σ'. Then, for each tree edges $A_{1-b}, n \overset{\ell_s}{\Rightarrow}_T A'_{1-b}, n'$, we apply the soundness of the partition tree (definition 16, item 5) and of the symbolic semantics (lemma 7 first point) and get the expected symbolic witness by induction hypothesis.

$\Leftarrow$ We construct a concrete witness w of (A_0^c, A_1^c). For each node of w_s labelled (B_0, B_1, n) with $B_i = (\mathcal{Q}_B^i, \mathcal{C}_B^i)$, w contains $(\mathcal{Q}_B^0\sigma_B, \Phi(\mathcal{C}_B^0)\sigma_B^0\!\downarrow), (\mathcal{Q}_B^1\sigma_B^1, \Phi(\mathcal{C}_B^1)\sigma_B\!\downarrow), \ell$, where we have $(f_{\mathsf{sol}}(n), \sigma_B^i) \in \mathit{Sol}(\mathcal{C})_B^i$ and ℓ is given by item 2 of definition 19. It is straightforward that w is a concrete witness by soundness of the partition tree (definition 16, item 5) and of the symbolic semantics (lemma 7 first point). Note in particular that static equivalence of processes in a same label of w follows from definition 15. $\square$

Finally, we can bound the size of the concrete witness of $A \not\approx_\ell B$.

Lemma 12. *There exists $p \in \mathbb{N}$ such that if P, Q are two plain processes, there exists $T \in$* $\mathsf{PTree}(P, Q)$ *such that if w_s is a symbolic witness of $(\{\!\{P\}\!\}, \mathcal{C}_\emptyset), (\{\!\{Q\}\!\}, \mathcal{C}_\emptyset), \mathit{root}(T)$ w.r.t. T such that $\mathit{Sol}(\mathsf{w}_s) \neq \emptyset$ then there exists a witness w_c of $(\{\!\{P\}\!\}, \emptyset) \not\approx_\ell (\{\!\{Q\}\!\}, \emptyset)$ such that $|\mathsf{w}_c| < 2^{|P,Q,\mathcal{R}|^p_{\mathsf{dag}}}$.*

Proof. It sufficies to prove that there exists a solution of w_s os size bounded by $2^{|P,Q,\mathcal{R}|^p_{\mathsf{dag}}}$ for some $p \in \mathbb{N}$ (where the size of a solution f_{sol} is $\sum_{N \in \mathit{dom}(f_{\mathsf{sol}})} |f_{\mathsf{sol}}(N)|$). Given P and Q, we first let T be a partition tree given by theorem 5 (whose mgs' are of exponential size). First, using the same reasoning as lemma 9 (i.e. definition 16 and lemma 7) we obtain a solution for each branch of w_s. The solution can then be built bottom-up, performing a unification at each node to ensure that all sibling nodes have father nodes mapped to a common solution. In particular, note that this unification preserves the exponential size of the solution by item 6 of definition 16. $\square$

Putting everything together we obtain:

Theorem 11. $\mathsf{Equiv}^{\approx_\ell}$ *for convergent subterm destructor rewriting systems is in co*NEXP.

Proof. By lemmas 11 and 12, if $P \not\approx_\ell Q$ then there exists a concrete witness w whose size is bounded by $2^{|P,Q,\mathcal{R}|^p_{DAG}}$ for some p. Hence we obtain a straightforward procedure that runs in exponential time on a non-deterministic machine. We first guess the witness w. Checking the validity of w requires to explore the possibly exponential number of branches of the tree and verify an exponential number of static equivalences. As the size of the frames is at most exponential (given the bound on the size of w) it follows from theorem 9, that the witnesses violating static equivalence of these frames can be checked in exponential time. Hence, this yields a NEXP procedure for checking whether $(\{\!\{P\}\!\}, \emptyset) \not\approx_\ell (\{\!\{Q\}\!\}, \emptyset)$. $\square$

Part III
Generating partition trees

8 Overview of the procedure

In the definition of partition trees of extended processes, the predicates π on second-order solutions of constraint systems occurring in the label of each node of the partition tree play a crucial role in the satisfaction of the properties stated in Definition 16. In particular, it is thank to these predicates restricting the solutions of constraint systems that we can partition the symbolic processes with statically equivalent solutions. In order to generate these predicates, we will introduce the notion of *extended constraint systems* that contain additional constraints allowing us to reason about the intruder knowledge. Intuitively, for each node n of the partition tree and for each symbolic process $(\mathcal{P},\mathcal{C}) \in \Gamma(n)$, our procedure will associate to $(\mathcal{P},\mathcal{C})$ an extended constraint systems $\mathcal{C}^e$ such that $Sol(\mathcal{C}^e) = Sol^{\pi(n)}(\mathcal{C})$.

Our procedure inductively builds the partition tree of $(\mathcal{P}_0,\mathcal{C}_\emptyset)$ and $(\mathcal{P}_1,\mathcal{C}_\emptyset)$ from the root down to the leaves. First, the procedure is initialized by associating two extended constraint systems $\mathcal{C}^e_0$ and $\mathcal{C}^e_1$ to $(\mathcal{P}_0,\mathcal{C}_\emptyset)$ and $(\mathcal{P}_1,\mathcal{C}_\emptyset)$ respectively such that $Sol(\mathcal{C}_\emptyset) = Sol(\mathcal{C}^e_i)$, for $i = 0, 1$. Second, we build the set $S_{root} = \{(\mathcal{P},\mathcal{C},\mathcal{C}^e) \mid (\mathcal{P}_i,\mathcal{C}_\emptyset,\mathcal{C}^e_i) \overset{\varepsilon}{\Rightarrow}_{\mathsf{s}} (\mathcal{P},\mathcal{C},\mathcal{C}^e), i = 0, 1\}$ where the transition $\xrightarrow{\varepsilon}_{\mathsf{s}}$ corresponds to the same transition rules in Figure 7 in which all constraints added to the constraint system $\mathcal{C}$ are also added to the extended constraint $\mathcal{C}^e$. Third, we will apply some *simplifications rules* on the extended constraint systems in S_{root} until we obtain a set S'_{root} such that $(\mathcal{P},\mathcal{C},\mathcal{C}^e_1) \in S'_{root}$ implies $(\mathcal{P},\mathcal{C},\mathcal{C}^e_2) \in S_{root}$ for some $\mathcal{C}^e_2$ and it is trivial to decide whether $Sol(\mathcal{C}^e_1)$ is empty or not. The root node of the partition tree is then defined as the node labeled $(\Gamma,\top,\varepsilon)$ where $\Gamma = \{(\mathcal{P},\mathcal{C}) \mid (\mathcal{P},\mathcal{C},\mathcal{C}^e) \in S'_{root} \wedge Sol(\mathcal{C}^e) \neq \emptyset\}$.

To build the children of the root node, we will apply all input and output symbolic transitions on the set S'_{root} previously constructed in order to obtain two new sets of extended symbolic processes S_{in} and S_{out} such that $S_{in} = \{(\mathcal{P}',\mathcal{C}',\mathcal{C}^{e\prime}) \mid (\mathcal{P},\mathcal{C},\mathcal{C}^e) \in S'_{root} \wedge (\mathcal{P},\mathcal{C},\mathcal{C}^e) \xRightarrow{Y(X)}_{\mathsf{s}} (\mathcal{P}',\mathcal{C}',\mathcal{C}^{e\prime})\}$ and $S_{out} = \{(\mathcal{P}',\mathcal{C}',\mathcal{C}^{e\prime}) \mid (\mathcal{P},\mathcal{C},\mathcal{C}^e) \in S'_{root} \wedge (\mathcal{P},\mathcal{C},\mathcal{C}^e) \xRightarrow{\overline{Z}\langle \mathsf{ax}_{n+1}\rangle}_{\mathsf{s}} (\mathcal{P}',\mathcal{C}',\mathcal{C}^{e\prime})\}$. Note that X, Y, Z are considered fresh and n is the size of the frames in the constraint systems in S'_{root}. At that point of the procedure, it still remains to partition the solutions of the constraint systems in order to obtain statically equivalent solutions. For that purpose, we will apply a series of *case distinction* and simplification rules on the extended constraint systems in S_{in} and S_{out} to obtain the sets of sets $\mathcal{S}_{in} = \{S^1_{in},\dots,S^p_{in}\}$ and $\mathcal{S}_{out} = \{S^1_{out},\dots,S^q_{out}\}$ such that for all $i \in \{1,\dots p\}$, if $(\mathcal{P},\mathcal{C},\mathcal{C}^e),(\mathcal{P}',\mathcal{C}',\mathcal{C}^{e\prime}) \in S^i_{in}$, $(\Sigma,\sigma) \in Sol(\mathcal{C}^e)$ then $(\Sigma,\sigma') \in Sol(\mathcal{C}^{e\prime})$ and $\Phi(\mathcal{C}^e)\sigma \sim \Phi(\mathcal{C}^{e\prime})\sigma'$; and similarly for $\mathcal{S}_{out}$. Note that this property corresponds to the property 2 of Definition 15 the symbolic processes in a node of the partition tree must satisfy. Hence the root node will have $p + q$ child defined as the nodes labeled (Γ,π,ℓ) if and only if:

- $\ell = Y(X)$, $i = \{1,\dots,p\}$, $\Gamma = \{(\mathcal{P},\mathcal{C}) \mid (\mathcal{P},\mathcal{C},\mathcal{C}^e) \in S^i_{in}\}$, π being the predicate such that $\Sigma \in \pi \Leftrightarrow \exists\sigma.(\Sigma,\sigma) \in Sol(\mathcal{C}^e)$ where $\mathcal{C}^e$ is an extended constraint systems in S^i_{in}.
- $\ell = \overline{Z}\langle \mathsf{ax}_{n+1}\rangle$, $i \in \{1,\dots,q\}$, $\Gamma = \{(\mathcal{P},\mathcal{C}) \mid (\mathcal{P},\mathcal{C},\mathcal{C}^e) \in S^i_{out}\}$, π being the predicate such that $\Sigma \in \pi \Leftrightarrow \exists\sigma.(\Sigma,\sigma) \in Sol(\mathcal{C}^e)$ where $\mathcal{C}^e$ is an extended constraint systems in S^i_{out}.

The procedure produces the children of these new nodes in a similar fashion starting this time with the sets S^i_{in} and S^j_{out} for all i, j.

Note that our case distinction and simplification rules will preserve the first order solutions of the extended constraint systems which will allow us to derive the property 3 of Definition 16 for partition tree. Moreover, the goals of distinction rules are twofold: They will partition the solutions of the extended constraint systems w.r.t. to the static equivalence as previously mentioned, but they will also simplify the constraint systems until it possesses a unique most general solution easily computed. The latter goal will allow us to prove all the properties on most general solutions in Definitions 15 and 16 that a partition tree must satisfy.

9 Extended constraint system

Before defining the extended constraint systems, we first need to extend our first-order logic for expressing constraints. More specifically, we introduce two new atomic formulas $\xi =^? \xi'$ and $\xi =^?_f \xi'$. The former formula is called *recipe equation* and its semantics correspond to the syntactic equality between the two recipes ξ, ξ', similarly to the syntactic equations on protocol terms. The latter formula is called equality fact and holds when the two recipes ξ and ξ' deduce the same protocol term w.r.t. a frame. Its semantics is defined as:

$$(\Phi, \Sigma, \sigma) \models \zeta =^?_f \zeta' \text{ iff } \zeta\Sigma\Phi{\downarrow} = \zeta'\Sigma\Phi{\downarrow},\ \mathit{Msg}(\zeta\Sigma\Phi) \text{ and } \mathit{Msg}(\zeta'\Sigma\Phi)$$

Moreover, we introduce particular kinds of first-order formulas called deduction, respectively equality formulas.

Definition 20 (Recipe, Deduction and equality formula). *A* recipe formula ψ *is a first-order logic formula built over recipe equations. A* deduction, respectively equality, formula *is defined as* $\forall S. H \Leftarrow C_1 \wedge \ldots \wedge C_n$ *where:*

- *S is a set of (both first order and second order) variables;*
- *H is a deduction, respectively equality, fact;*
- *for all $i \in \{1, \ldots, n\}$, C_i is either a a deduction fact of the form $X \vdash^? t$ with $X \in \mathcal{X}^2$ or a syntactic equation.*

A deduction, resp. equality, formula ψ is solved *when*

$$\psi = \forall X_1, \ldots X_n, x_1, \ldots, x_n. H \Leftarrow X_1 \vdash^? x_1 \wedge \ldots \wedge X_n \vdash^? x_n$$

and all X_i, x_i $(1 \leq i \leq n)$ are pairwise distinct variables, $\{X_1, \ldots X_n\} \subseteq vars(H)$. When H is a deduction fact, its protocol term is called the head term *of the formula.*

We note that in particular any deduction, respectively equality, fact is a solved deduction, respectively equality, formula with no bound variables.

Given a formula $\psi = (\forall S. H \Leftarrow \varphi)$, we denote by $\mathsf{D}(\varphi)$ the set of deduction facts in φ, $\mathsf{E}^1(\varphi)$ the set of formulas over syntactic equations in φ. We denote $\alpha[\psi] = mgu(\mathsf{E}^1(\varphi))$ and we also denote by $vars(\psi)$, $vars^1(\psi)$, $vars^2(\psi)$ the free (respectively, first-order, or second-order) variables of ψ and by $bvars(\psi)$, $bvars^1(\psi)$, $bvars^2(\psi)$ the bound (respectively, first-order, second-order) variables of ψ.

Intuitively, a deduction formula represents which terms are deducible by the adversary. The premisses $C_1, \ldots, C_n$ express conditions under which the deduction fact holds. For solved deduction facts these premisses are trivially satisfiable. Similarly, equality formulas express which recipes are equivalent, i.e., allow the adversary to deduce the same term. As we are interested in equivalence properties, we need to keep both track of deducible terms, and equivalence of recipes.

We now introduce the notion of consequence, which defines the set of terms derivable by an adversary by applying public contexts to already deducible terms described by a set of solved deduction facts.

Definition 21 (Consequence)**.** *Let* S *be a set of deduction facts. We define the* set of consequences of S*, denoted* $\textsc{Conseq}(\mathsf{S})$ *as the set of pair* $(C[\xi_1, \ldots, \xi_n], C[u_1, \ldots, u_n])$ *such that* $C[\_, \ldots, \_]$ *is a context built on* $\mathcal{F}_c \cup \mathcal{N}_{\mathsf{pub}}$ *and for all* $i \in \{1, \ldots, n\}$*,* $\xi_i \vdash^? u_i \in \mathsf{S}$*.*

We write $\xi \in \textsc{Conseq}(\mathsf{S})$*, if* $\exists t.(\xi, t) \in \textsc{Conseq}(\mathsf{S})$*.*

If $\xi \in \textsc{Conseq}(\mathsf{S})$ with $\xi = C[\xi_1, \ldots, \xi_n]$ as described in Definition 21, we denote by $st_c(\xi, \mathsf{S})$ the set $\{\xi|_p \mid p \text{ is a position of } \mathcal{C}[\_, \ldots, \_]\}$. If R is a set of recipes, we denote by $st_c(R, \mathsf{S})$ the set $\cup_{\xi \in R} st_c(\xi, \mathsf{S})$.

We finally define the notion of *extended constraint systems* and their solutions.

Definition 22. *An extended constraint system is* $\mathcal{C}$ *defined as* $(\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ *where:*

- $(\Phi, \mathsf{D}, \mathsf{E}^1)$ *is a constraint system;*
- E^2 *is a conjunction of recipe equations or formulas of the form* $\forall Y_1, \ldots, Y_k. \bigvee_{j=1}^{p} \xi_j \neq^? \zeta_j$*;*
- K *is a conjunction of deduction facts;*
- F *is a conjunction of deduction or equality formulas.*

Additionally, we ask that $vars^1(\mathsf{K}, \mathsf{F}) \subseteq vars^1(\mathsf{E}^1, \mathsf{D})$*,* $vars^2(\mathsf{K}, \mathsf{F}) \subseteq vars^2(\mathsf{E}^2, \mathsf{D})$ *and all universally quantified second-order variables of* E^2*,* K*,* F *are in* $\mathcal{X}^2_{|\Phi|}$*. Finally, we require that there exists* $\sigma = mgu(\mathsf{E}^1|_=)$ *such that* $vars^1(img(\sigma)) \subseteq vars^1(\mathsf{D})$*.*

The set E^2 adds new constraints on recipes to be fulfilled and the set of formulas in K and F represent the attacker knowledge. The solved deduction facts in K represent the deducible protocol terms: a deduction fact $\xi \vdash^? u$ expresses that the attacker knows u by using the recipe ξ. The set of deduction formulas in F reasons about potentially deducible terms. After an output transition, our algorithm will apply some case distinctions on whether the new output term can generate new protocol terms deducible by the attacker. The conditions under which such terms are deducible are modelled by adding a deduction formula in F. The algorithm will try to simplify this deduction formula until it becomes a deduction fact. When it is the case, it can be easily decided whether the deduction formula's head protocol term is in fact a new deducible term (it will then be added to K) or a consequence of the deduction facts already in K. Similarly equality formulas represent potential equalities that may hold.

Definition 23 (Solutions)**.** *A pair of substitutions* (Σ, σ) *is a solution of an extended constraint system* $(\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ *if* $(\Phi\sigma, \Sigma, \sigma) \models \mathsf{D} \wedge \mathsf{E}^1 \wedge \mathsf{E}^2$ *and the following two properties hold:*

- *for all* $\xi \in st(img(\Sigma)) \cup sst(\mathsf{K}\Sigma)$*,* $Msg(\xi\Phi\sigma)$ *and* $(\xi, \xi\Phi\sigma\!\downarrow) \in \textsc{Conseq}(\mathsf{K}\Sigma\sigma)$
- *for all* $\xi, \xi' \in st_c(img(\Sigma), \mathsf{K}\Sigma)$*,* $\xi\Phi\sigma\!\downarrow = \xi'\Phi\sigma\!\downarrow$ *implies* $\xi = \xi'$*.*

The set of solutions of an extended constraint system $\mathcal{C}$ *is denoted* $Sol(\mathcal{C})$*. An extended constraint system* $\mathcal{C}$ *is satisfiable if* $Sol(\mathcal{C}) \neq \emptyset$*. We will denote by* $\perp$ *an unsatisfiable extended constraint system.*

10 Computing most general solutions of an extended constraint system

From the definition, we know that the recipes in a solution of an extended constraint system must always be consequence of set of deduction facts K, i.e. they should always be built as a public context applied to recipe of K. Moreover, we also know that two different recipes in the solution should deduce different protocol terms. Thanks to these properties, the most general solutions of an extended constraint system can be easily computed with a simple constraint solving transition system. For instance, the uniformity of a solution can be expressing by the following transition rule.

$$(\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \quad \xrightarrow{\Sigma} \quad (\Phi, \mathsf{D}', \mathsf{E}^1, \mathsf{E}^2\Sigma \wedge \Sigma, \mathsf{K}\Sigma, \mathsf{F}\Sigma) \tag{8}$$

if $(\xi, \zeta) \in \mathsf{R}(\mathcal{C})^2 \cup (\mathcal{N}_{\mathsf{pub}} \times vars^2(\mathsf{D}))$ and $\exists u.\ (\xi, u), (\zeta, u) \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$, $\Sigma = mgu(\xi =^? \zeta)$, $\Sigma \neq \emptyset$, $\Sigma \neq \perp$, $\mathsf{D}' = \mathsf{D} \setminus \{X \vdash^? v \in \mathsf{D} \mid X \in dom(\Sigma)\}$ and $\mathsf{R}(\mathcal{C})$ is syntactic sugar for $st_c(img(mgu(\mathsf{E}^2(\mathcal{C}))), \mathsf{K} \cup \mathsf{D}) \cup vars^2(\mathsf{D})$.

Note that the set $\mathsf{R}(\mathcal{C})$ typically represents all recipes in the constraint system that are already used to constraint the solutions of $\mathcal{C}$. Therefore, when u is not a public name or a term already deduced by a recipe in $\mathsf{R}(\mathcal{C})$, we can apply one of the two following rules, defined as follows, indicating that either the solutions are rooted by a public symbol or are recipes of K.

$$(\Phi, \mathsf{D} \wedge X{:}k \vdash^? u, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \quad \xrightarrow{\Sigma} \quad (\Phi, \mathsf{D} \wedge \bigwedge_{i=1}^{n} X_i{:}k \vdash^? u_i, \mathsf{E}^1, \mathsf{E}^2\Sigma \wedge \Sigma, \mathsf{K}\Sigma, \mathsf{F}\Sigma) \tag{9}$$

if $u = \mathsf{f}(u_1, \ldots, u_n)$, $\Sigma = \{X \to \mathsf{f}(X_1, \ldots, X_n)\}$ with $X_1, \ldots, X_n$ fresh and for all $\xi, \zeta \in \mathsf{R}(\mathcal{C}) \setminus \{X\}$, $(\xi, u) \notin \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$,

$$(\Phi, \mathsf{D} \wedge X{:}k \vdash^? u, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \quad \xrightarrow{\Sigma} \quad (\Phi, \mathsf{D}, \mathsf{E}^1 \wedge u =^? v, \mathsf{E}^2\Sigma \wedge \Sigma, \mathsf{K}\Sigma, \mathsf{F}\Sigma) \tag{10}$$

when $u \notin \mathcal{X}^1$, $\xi \vdash^? v \in \mathsf{K}$, $mgu(X =^? \xi) \neq \perp$, $\Sigma = \{X \to \xi\}$ and for all $\xi \in \mathsf{R}(\mathcal{C}) \setminus \{X\}$, $(\xi, u) \notin \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$.

We also simplify the syntactic equations and recipes equations on the constraint systems after each transition step, e.g. by computing the mgu of the terms u and v in Rule 10 and applying it on the rest of the constraint system, or by replacing the constraint system with $\perp$ when u and v are not unifiable. Theses simplifications typically apply the equations on the constraint systems and ensure that the disequations and uniformity property are not trivially unsatisfiable. The simplification rules in Figure 11 and as follows.

$$(\Phi, \mathsf{D}, \mathsf{E}^1 \wedge x =^? u, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \quad \rightsquigarrow \quad (\Phi\sigma, \mathsf{D}\sigma, \mathsf{E}^1\sigma \wedge x =^? u, \mathsf{E}^2, \mathsf{K}\sigma, \mathsf{F}\sigma) \tag{11}$$
$$\text{if } x \notin vars^1(u), x \in vars^1(\mathsf{E}^1, \mathsf{D}, \Phi, \mathsf{K}, \mathsf{F}) \text{ and } \sigma = \{x \to u\}$$

$$(\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \quad \rightsquigarrow \quad \bot \tag{12}$$
$$\text{if } \xi, \zeta \in \mathsf{R}(\mathcal{C}),\ (\xi, u), (\zeta, u) \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D}) \text{ and } (\mathsf{E}^2 \wedge \xi =^? \zeta)\wr = \bot$$

$$\neg\top \rightsquigarrow \bot \qquad \neg\bot \rightsquigarrow \top \qquad \varphi \vee \top \rightsquigarrow \top \qquad \varphi \vee \bot \rightsquigarrow \varphi \qquad \pi =^? \pi \rightsquigarrow \top$$

$$\begin{array}{rcll}
\forall S \cup \{\alpha\}.\ \varphi & \rightsquigarrow & \forall S.\ \varphi & \text{if } \alpha \notin vars(\varphi) \\
\bigvee_{i=1}^{n} \pi_i \neq^? \varpi_i & \rightsquigarrow & \top & \text{if } mgu(\bigwedge_{i=1}^{n} \pi_i =^? \varpi_i) = \bot \\
\mathsf{f}(\pi_1, \ldots, \pi_n) =^? \mathsf{f}(\varpi_1, \ldots, \varpi_n) & \rightsquigarrow & \bigwedge_{i=1}^{n} \pi_i =^? \varpi_i & \\
\forall S \cup \{\alpha\}.(\varphi \vee \alpha \neq^? \pi) & \rightsquigarrow & \forall S.\varphi\sigma & \text{if } \sigma = \{\alpha \to \pi\} \text{ and } \alpha \notin vars(\pi) \\
\forall S.(\varphi \vee \alpha \neq^? \pi) & \rightsquigarrow & \forall S.(\varphi\sigma \vee \alpha \neq^? \pi) & \text{if } \sigma = \{\alpha \to \pi\} \text{ and } \alpha \in vars(\varphi) \setminus vars(\pi)
\end{array}$$

Figure 11: Simplification rules on formulae

The simplification rules displayed in Figure 11 apply to formulas in $\mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{F}$. We present the simplification rules, as if all formulae were in disjunctive normal form. As some rules are similar for both protocol terms and recipes we use metavariables (α, π, ϖ) which can be either replaced by (x, u, v) or $(X{:}i, \xi, \zeta)$ where $\xi, \zeta \in \mathcal{T}_i^2$. The first block of rules reflect standard logical rules. The second block of rules simplify (recipe) (dis)equations which may appear in $\mathsf{E}^1, \mathsf{E}^2, \mathsf{F}$. The third block of rules applies a substitution $\{\alpha \mapsto \pi\}$ in case we have a disjunct $\alpha \neq^? \pi$. Note that when α is bound the disjunct may be completely removed while otherwise it is essential for the correctness of the simplification rule to keep the disjunct (as α appears in the domain of the solution). The simplification relation is homomorphically extended to more complex formulas, i.e., $\varphi[\psi] \rightsquigarrow \varphi[\psi']$ if $\psi \rightsquigarrow \psi'$. Note that our simplification rules are confluent modulo renaming of variables. Hence, we denote $\mathcal{C}\wr$ the extended constraint system such that $\mathcal{C} \rightsquigarrow^* \mathcal{C}\wr$ and $\mathcal{C}\wr \not\rightsquigarrow$.

To obtain the most general unifier of the constraint system, it suffices to apply the transition rules until none is applicable. In such a case, we consider that the constraint system in *solved form*. In particular, it implies that all deduction facts in D have distinct variables as right hand term and for all $(\xi, \zeta) \in \mathcal{R}(\mathcal{C})^2 \cup (\mathcal{N}_{\mathsf{pub}} \times vars^2(\mathsf{D}))$, $(\xi, u), (\zeta, u) \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$ implies $\xi = \zeta$. For such constraint system, we can easily show that $mgu(\mathsf{E}^2)\Sigma$ is a solution where Σ is a bijection from the recipes variables in D to fresh public names, meaning that its most general solutions are unique and is the substitution $mgu(\mathsf{E}^2)$. Let us we denote $\mathcal{C} \xrightarrow{\Sigma}\wr\, \mathcal{C}'$ when $\mathcal{C} \xrightarrow{\Sigma} \mathcal{C}''$ and $\mathcal{C}' = \mathcal{C}''\wr$, and $\mathcal{C} \overset{\Sigma}{\Rightarrow}\wr\, \mathcal{C}'$ when $\mathcal{C} \xrightarrow{\Sigma_1}\wr \ldots \xrightarrow{\Sigma_n}\wr\, \mathcal{C}'$ with $\Sigma = \Sigma_1 \ldots \Sigma_n$, we prove the following lemma.

Lemma 13. *Let $\mathcal{C}$ be an extended constraint system obtained during the procedure*

$$mgs(\mathcal{C}) = \{\Sigma|_{vars^2(\mathcal{C})} \mid \mathcal{C} \overset{\Sigma}{\Rightarrow}\wr\, \mathcal{C}', \mathcal{C}' \neq \bot, \mathcal{C}' \text{ solved}\}$$

11 Symbolic rules

As mentioned in the overview, our procedure will alternate between symbolic, simplification and case distinction rule. In Figure 7, we provided the symbolic rules for a process and its constraint system. The symbolic rules for a process, its contraint system and extended constraint system are very similar as they add the same contraint on both contraint system and extended constraint system (see Figure 12).

$(\mathcal{P} \cup \{\!\{0\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P}, \mathcal{C}, \mathcal{C}^e)$ (E-NULL)

$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{\varepsilon}_{\mathsf{s}}$ (E-THEN)
$(\mathcal{P} \cup \{\!\{Q_1\}\!\}, \mathcal{C}[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma], \mathcal{C}^e[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma])$
if $\sigma \in mgu_{\mathcal{R}}(u\mu\!\downarrow =^? v\mu\!\downarrow)$

$(\mathcal{P} \cup \{\!\{\mathsf{if}\ u = v\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{\varepsilon}_{\mathsf{s}}$ (E-ELSE)
$(\mathcal{P} \cup \{\!\{Q_2\}\!\}, \mathcal{C}[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge syn_{\mathcal{R}}(u\mu\!\downarrow \neq^? v\mu\!\downarrow)], \mathcal{C}^e[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge syn_{\mathcal{R}}(u\mu\!\downarrow \neq^? v\mu\!\downarrow)])$

$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.Q_1, v(x).Q_2\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{\varepsilon}_{\mathsf{s}}$ (E-COMM)
$(\mathcal{P} \cup \{\!\{Q_1, Q_2\{x \to t\}\}\!\}, \mathcal{C}[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma], \mathcal{C}^e[\mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma])$
if $\sigma \in mgu_{\mathcal{R}}(u\mu\!\downarrow =^? v\mu\!\downarrow, t\mu\!\downarrow =^? t\mu\!\downarrow)$

$(\mathcal{P} \cup \{\!\{P \mid Q\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{\varepsilon}_{\mathsf{s}} (\mathcal{P} \cup \{\!\{P, Q\}\!\}, \mathcal{C}, \mathcal{C}^e)$ (E-PAR)

$(\mathcal{P} \cup \{\!\{u(x).Q\}\!\}, \mathcal{C}, \mathcal{C}^e) \xrightarrow{Y(X)}_{\mathsf{s}}$ (E-IN)
$(\mathcal{P} \cup \{\!\{Q\}\!\}, \mathcal{C}[\mathsf{D} \mapsto \mathsf{D} \wedge X \vdash^? x \wedge Y \vdash^? y, \mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma],$
$\mathcal{C}^e[\mathsf{D} \mapsto \mathsf{D} \wedge X \vdash^? x \wedge Y \vdash^? y, \mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma])$
if $\sigma \in mgu_{\mathcal{R}}(y =^? u\mu\!\downarrow)$ and $X, Y \in \mathcal{X}^2_{:|\Phi|}$

$(\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.Q\}\!\}, (\Phi, \mathsf{D}, \mathsf{E}^1), \mathcal{C}^e) \xrightarrow{\overline{Y}\langle \mathsf{ax}_n\rangle}_{\mathsf{s}}$ (E-OUT)
$(\mathcal{P} \cup \{\!\{Q\}\!\}, (\Phi \cup \{\mathsf{ax}_n \mapsto t\sigma\!\downarrow\}, \mathsf{D} \wedge Y \vdash^? y, \mathsf{E}^1 \wedge \sigma),$
$\mathcal{C}^e[\Phi \mapsto \Phi \cup \{\mathsf{ax}_n \mapsto t\sigma\!\downarrow\}, \mathsf{D} \mapsto \mathsf{D} \wedge Y \vdash^? y, \mathsf{E}^1 \mapsto \mathsf{E}^1 \wedge \sigma])$
if $\sigma \in mgu_{\mathcal{R}}(y =^? u\mu\!\downarrow, t\mu\!\downarrow =^? t\mu\!\downarrow)$, y is fresh, $X \in \mathcal{X}^2_{:n}$ and $n = |\Phi| + 1$

where u, v and t are terms, $x \in \mathcal{X}^1$, $X, Y \in \mathcal{X}^2$, $\mathsf{ax}_n \in \mathcal{AX}$ and $\mu = mgu(\mathsf{E}^1(\mathcal{C})|_=)$.

Figure 12: Symbolic semantics with extended constraint systems

12 Simplification rules

Before defining the case distinction rules, we consider additional *simplification rules* to the one defined for computing most general solutions and displayed in 13 and 14. These rules aim to simplify the set of symbolic extended processes generated by the algorithm so that they can reach a simple form. In particular, a set of symbolic extended processes in simple form will have the remarkable property of having all its extended constraint systems pairwise symbolically equivalent. Such a simple form is reached when for all extended constraint systems $\mathcal{C}$, the head terms of $\mathsf{D}(\mathcal{C})$'s constraints are pairwise distinct variables, $\mathsf{E}^1(\mathcal{C})$ only contains equations of the form $x_i =^? u_i$ which can be interpreted as a (acyclic) substitution $\{x_i \to u_i\}_i$ such that the variables in its domain do not occur anywhere else in $\mathcal{C}$. Moreover, a constraint system in simple form also have saturated sets $\mathsf{K}(\mathcal{C})$ and $\mathsf{F}(\mathcal{C})$, i.e., all messages that are deducible by the attacker are a consequence of $\mathsf{K}(\mathcal{C})$ and $\mathsf{D}(\mathcal{C})$, and all equalities of recipes within the frame are instances of a solved equality formula in $\mathsf{F}(\mathcal{C})$.

The simplification rules in Figure 13 directly focus on extended constraint systems and are naturally uplifted to (sets of (sets of)) extended symbolic processes. We also naturally lift the rules of Figure 11 from individual elements of the extended constraint system to the extended constraint system itself. For example, $(\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \rightsquigarrow (\Phi, \mathsf{D}, \mathsf{E}^{1\prime}, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ if $\mathsf{E}^1 \rightsquigarrow \mathsf{E}^{1\prime}$.

$$(\Phi, \mathsf{D}, \mathsf{E}^1 \wedge x =^? u, \mathsf{E}^2, \mathsf{K}, \mathsf{F}) \rightsquigarrow (\Phi\sigma, \mathsf{D}\sigma, \mathsf{E}^1\sigma \wedge x =^? u, \mathsf{E}^2, \mathsf{K}\sigma, \mathsf{F}\sigma) \quad (13)$$
$$\text{if } x \notin \mathit{vars}^1(u), x \in \mathit{vars}^1(\mathsf{E}^1, \mathsf{D}, \Phi, \mathsf{K}, \mathsf{F}) \text{ and } \sigma = \{x \to u\}$$

$$\mathcal{C} \rightsquigarrow \bot \qquad \text{if } \mathit{mgs}(\mathcal{C}) = \emptyset \quad (14)$$

$$\mathcal{C}[\mathsf{E}^1 \wedge \forall \tilde{x}.\phi] \rightsquigarrow \mathcal{C}[\mathsf{E}^1 \wedge \forall \tilde{x}.\phi \mapsto \mathsf{E}^1] \quad \text{if } \mathit{mgs}(\mathcal{C}[\mathsf{E}^1]\mathit{mgu}(\forall \tilde{x}.\phi)) = \emptyset \quad (15)$$

$$\mathcal{C}[\mathsf{F} \cup \{\psi\}] \rightsquigarrow \mathcal{C}[\mathsf{F} \cup \{\psi\} \mapsto \mathsf{F}] \quad \text{if } \mathit{mgs}(\mathcal{C}|^{\psi}) = \emptyset \quad (16)$$

$$\mathcal{C}[\mathsf{F} \cup \{\psi\}] \rightsquigarrow \mathcal{C}[\mathsf{F} \cup \{\psi\} \mapsto \mathsf{F}] \quad (17)$$
$$\text{if } \psi' \in \mathsf{F},\ \psi' \simeq_{\mathsf{r}} \psi \text{ and } \psi' \text{ solved}$$

Figure 13: Normalisation rules on constraint systems

Rule 13 propagates a simple constraint of the form $x =^? u$ in E^1 to the whole constraint system. The next three rules exploit the existence of a most general solution of the constraint system. In particular Rule 14 checks whether the constraint system is unsatisfiable, i.e. does not have most general solution. When it is the case then the constraint system is transform into $\bot$. Rule 15 similarly remove a disequations $\forall \tilde{x}.\phi$ in E^1 when she is trivially satisfiable. In particular, the rule will check whether the constraint system $\mathcal{C}$ without the disequation $\forall \tilde{x}.\phi$, i.e. $\mathcal{C}[\mathsf{E}^1]$, can have a solution that would contradict the disequation. To enforce such solutions, we apply on $\mathcal{C}[\mathsf{E}^1]$ the substitution $\mathit{mgu}(\forall \tilde{x}.\phi)$ where $\mathit{mgu}(\forall \tilde{x}. \bigvee_i u_i \neq^? v_i)$ is the substitution $\mathit{mgu}(\bigwedge_i u_i =^? v_i)$ with a fresh renaming of the variables in $\tilde{x}$.

Finally Rule 16 remove a formula from F when its hypotheses are unsatisfiable in the constraint system. To check this, we add the hypothesis to the constraint system and verifies the existence of most general solutions. Formally, we given a constraint system $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ and a formula $\psi = \forall S.H \Leftarrow \varphi$, we define $\mathcal{C}|^{\psi}$ as the constraint system $(\Phi, \mathsf{D} \cup \mathsf{D}(\psi), \mathsf{E}^1 \wedge \mathsf{E}^1(\psi), \mathsf{E}^2, \mathsf{K}, \mathsf{F})$.

Rule 17 removes an unsolved deduction or equality formula ψ from F when there exists another recipe equivalent, solved formula. Formally, two deduction, respectively equality formulas ψ and ψ' are *recipe equivalent*, denoted $\psi \simeq_{\mathsf{r}} \psi'$, when they have the same recipes (up to α-conversion of bound variables) in the head of the formula and put the same constraints on the recipes in the hypotheses of the formulas. Typically, the solved, recipe equivalent formula subsumes all the solutions of ψ.

We also consider normalization rules that focus on sets of sets of symbolic extended processes (Figure 14). Rule 18 allows to remove $\bot$ elements of the vector. In rules 20 and 21, we assume that $\forall 1 \leq i \leq n.\ \mathcal{C}^e_i = (\Phi_i, \mathsf{D}_i, \mathsf{E}^1_i, \mathsf{E}^2_i, \mathsf{K}_i, \mathsf{F}_i)$, $\mathcal{C}^e_i \not\rightsquigarrow$ and $\mathcal{C}^e_i \neq \bot$. Rule 19 split a set of extended symbolic processes whenever a common solution would yield statically inequivalent frames. More specifically, the rule separates constraint systems in which a given recipe always yields a message, resp. an equality always holds, from the constraint systems in which the same recipe would never yield a message, resp. the same equality would never

$$\mathcal{S} \cup \{\Gamma \cup \{(\mathcal{P}, \mathcal{C}, \bot)\}\} \rightsquigarrow \mathcal{S} \cup \{\Gamma\} \tag{18}$$

$$\mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n \cup \{(\mathcal{P}'_i, \mathcal{C}'_i, \mathcal{C}_i^{e\prime})\}_{i=1}^m\} \rightsquigarrow \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n, \{(\mathcal{P}'_i, \mathcal{C}'_i, \mathcal{C}_i^{e\prime})\}_{i=1}^m\} \tag{19}$$

if $n \neq 0$, $m \neq 0$ and there exists ψ s.t. $\forall i$, $\exists \psi' \in \mathsf{F}(\mathcal{C}_i^e).\psi \simeq_{\mathsf{r}} \psi'$ and ψ' solved; and $\forall i, \forall \psi' \in \mathsf{F}(\mathcal{C}_i^{e\prime})$, $\psi \not\simeq_{\mathsf{r}} \psi'$.

$$\mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n\} \rightsquigarrow \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{K} \mapsto \mathsf{K} \cup \{\psi_i\}])\}_{i=1}^n\} \tag{20}$$

if $\forall i.\ \mathcal{C}_i^e$ is solved and $\exists \xi.\forall i.\exists u_i.\ \psi_i = \xi \vdash^? u_i \in \mathsf{F}_i$ and $u_i \notin \textsc{Conseq}(\mathsf{K}_i \cup \mathsf{D}_i)$.

$$\mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n\} \rightsquigarrow \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{F} \mapsto \mathsf{F} \cup \{\psi{:}(\Sigma, \mathcal{C}_i)\}])\}_{i=1}^n\} \tag{21}$$

if $\forall i.\ \mathcal{C}_i$ is solved, $\exists \xi.\forall i.\exists u_i.\ \xi \vdash^? u_i \in \mathsf{F}_i$ and $\exists i_1.\zeta$ such that:

- $(\zeta, u_1) \in \textsc{Conseq}(\mathsf{K}_{i_1} \cup \mathsf{D}_{i_1})$
- $\Sigma = \{X \to \xi, Y \to \zeta\}$ and $\psi = \forall \{X, Y, z\}.(X =_f^? Y) \Leftarrow X \vdash^? z \wedge Y \vdash^? z$ with X, Y, z fresh variables.
- $\forall i.\forall(\forall S.\zeta_1 =_f^? \zeta_2 \Leftarrow \varphi) \in \mathsf{F}_i$, $\zeta_1 \neq \xi$ or $\zeta_2 \neq \xi$

Figure 14: Simplification rules for sets of sets of extended symbolic processes

hold. This is characterised by the fact that a deduction, resp. equality, formula is solved in some constraint systems and not in the others. Rule 20 allows to add a solved deduction formula from F_i to K_i when this formula is solved in all extended constraint systems of the set and the first-order terms in their head are not already a consequence of the set of solved deduction formulae. Finally rule 21 adds an equality formula to each constraint system when the protocol term in a head's solved deduction fact is a consequence of its set of solved deduction facts K, modeling the fact that $\xi =_f^? \zeta$ should hold. For termination purposes, we also require that at least one recipe equivalent deduction fact in some other constraint systems should not be consequence of its set of solved deduction facts.

Note that in rule 21, we rely on a new notation $\psi{:}(\Sigma, \mathcal{C}_i)$ to describe the formula generated by applying the substitution Σ on the formula ψ. However, such application is different from the classical application of substitution uplifted to formula. More specifically, we will ensure that all second order variables are linked to a deduction fact in ψ. This is formally defined as follows.

Definition 24. *Let $C = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ be an extended constraint system. Let $\psi = \forall S.H \Leftarrow \varphi$ be a formula. Let Σ be a substitution. We denote $\psi{:}(\Sigma, \mathcal{C})$ the formula*

$$\forall S'.H\Sigma \Leftarrow \mathsf{E}^1(\psi) \wedge D \wedge E$$

where $D = \mathsf{D}(\psi) \cup D'$, $D' = \{Y \vdash^? y \mid Y \in vars^2(img(\Sigma)) \setminus vars^2(\mathcal{C}, \psi), y\ fresh\} \setminus \{Y \vdash^?$

$u \in \mathsf{D}(\psi) \mid Y \in dom(\Sigma)\}$, $S' = vars^1(D')$ and $E = \{u =^? v \mid Y \vdash^? u \in \mathsf{D}(\psi), Y \in dom(\Sigma), (Y\Sigma, v) \in$ CONSEQ$(\mathsf{K} \cup \mathsf{D} \cup D)$.

Note that our simplifications are not confluent in general. Thus we denote by $\mathcal{S}\wr$ the set $\{\mathcal{S}' \mid \mathcal{S} \rightsquigarrow^* \mathcal{S}' \wedge \mathcal{S}' \not\rightsquigarrow\}$. We call each element of $\mathcal{S}\wr$ an irreducible form of $\mathcal{S}$.

13 Case distinction rules

As previously mentioned, the purposes of case distinction rules is to transform a set of extended symbolic process into sets of extended symbolic process where the constraint systems in each set have statically equivalence solutions. As such, one can see our case distinction rules as transition rules on a set of sets of extended symbolic process. We consider three case distinction rules and one final rule.

13.1 Rule SAT

The first rule, called SAT, focuses on the satisfiability of the extended constraint systems. Intuitively, since we aim to obtain that each constraint systems in a set has the same second-order solution, they should also have the same most general solutions. Therefore, for all most general solutions of a constraint system in a set, the rule will do a case distinction on whether or not the solutions of all the constraint systems in the same set are an instance of this most general solution. This can be formalized as follows. The satisfiability case distinction rule will also be applied to solve syntactic disequations and formulas in F. The rule SAT can be formalized as follows.

$$\mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n\} \rightarrow \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e{:}\Sigma)\}_{i=1}^n\} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{E}^2 \wedge \neg\Sigma])\}_{i=1}^n\} \qquad (22)$$

if there exists j such that either (a) $\mathcal{C}_j^e$ not solved and $\Sigma \in mgs(\mathcal{C}_j^e)$ or (b) $\exists \psi \in \mathsf{F}(\mathcal{C}_j^e)$. ψ not solved and $\Sigma \in mgs(\mathcal{C}_j^e|^{\psi})$ or (c) $\exists (\forall \tilde{x}.\phi) \in \mathsf{E}^1(\mathcal{C}_j^e)$. $\Sigma \in mgs(\mathcal{C}_j^{e\prime} mgu(\forall \tilde{x}.\phi))$ where $\mathcal{C}_j^{e\prime}$ is the constraint system $\mathcal{C}_j^e$ where $\forall \tilde{x}.\phi$ was removed from $\mathsf{E}^1(\mathcal{C}_j^e)$.

Note that $\neg\Sigma$ is syntactic sugar for $\forall S. \bigvee_{X \in dom(\Sigma)} X \neq^? X\Sigma$ with $S = vars^2(\Sigma) \setminus vars^2(\mathcal{C}_i^e)$. Moreover, $\mathcal{C}{:}\Sigma$ denotes the application of the most general solution Σ on $\mathcal{C}$. Similarly to the application of most general solutions on a formula defined in Definition 24, $\mathcal{C}{:}\Sigma$ is not the standard application of a substitution on a constraint system. Indeed, all recipe variables in a constraint system must either appear in a deduction fact in D or in the domain of $mgu(\mathsf{E}^2)$. Hence, all the new recipe variables that can be introduced by the most general solution Σ should be affected to a deduction fact. For example, if $\Sigma = \{X \rightarrow \mathsf{f}(\xi, X_1, X_2)\}$ with X_1, X_2 new variables and $\mathcal{C} = (\Phi, \mathsf{D} \wedge X{:}k \vdash^? u, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ with $\xi \vdash^? v \in \mathsf{K}$ then $\mathcal{C}{:}\Sigma$ is the constraint system $(\Phi, \mathsf{D} \wedge X_1 \vdash^? x_1 \wedge X_2 \vdash^? x_2, \mathsf{E}^1 \wedge u =^? \mathsf{f}(v, x_1, x_2), \mathsf{E}^2\Sigma, \mathsf{K}\Sigma, \mathsf{F}\Sigma)$. This is formally defined as follows.

Definition 25. *Let $C = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ be an extended constraint system. Let Σ be a substitution. We denote $\mathcal{C}{:}\Sigma$ the constraint system:*

$$(\Phi, D, E, \mathsf{E}^2\Sigma \wedge \Sigma|_{vars^2(\mathcal{C})}, \mathsf{K}\Sigma, \mathsf{F}\Sigma)$$

where $D = \mathsf{D} \cup \{Y \vdash^? y \mid Y \in vars^2(img(\Sigma|_{vars^2(\mathcal{C})})) \setminus vars^2(\mathcal{C}), y\ fresh\} \setminus \{Y \vdash^? u \in \mathsf{D} \mid Y \in dom(\Sigma)\}$ and $E = \mathsf{E}^1 \cup \{u =^? v \mid X \vdash^? u \in \mathsf{D}, X \in dom(\Sigma), (X\Sigma, v) \in$ CONSEQ$(\mathsf{K}\Sigma \cup D)\}$.

13.2 Rule REW

The second case distinction rule focuses on the knowledge of the attacker. For example, when the last symbolic transition corresponds to an output, the rule will apply rewrite rules on this output to determine whether new messages can be learned by the attacker. Generally speaking, since deduction facts in K represent the current knowledge of the intruder, the rule Rew checks whether an attacker can apply a rewrite rule $\ell \to r$ on a deduction fact $\xi_0 \vdash^? u_0$ from the set of deduction facts K in a constraint system to deduce a new term. For the rewrite rule to apply the attacker may however need to first apply a context on u_0. To define the context that unifies with the left-hand side of the rewrite rule and, at position p, with the term u_0 we introduce the notion of a *skeleton*.

Definition 26. *Given a term ℓ and a position p of t we define the set of* skeletons for (t,p), *denoted* $\mathsf{Skel}(t,p)$, *to be the set of tuple (ξ, u, D) such that $\xi \in \mathcal{T}(\mathcal{F}, \mathcal{X}^2)$, $u \in \mathcal{T}(\mathcal{F}, \mathcal{X}^1)$, D is a set of deduction facts and*

$$(root(\xi|_q), root(u|_q)) = \begin{cases} (root(t|_q), root(t|_q)) & \textit{for any strict prefix } q \textit{ of } p \\ (X_q, x_q) & \textit{for any other position } q \textit{ of } \xi \end{cases}$$

where the set of variables X_q (resp. x_q) (q a position of ξ) are fresh, pairwise distinct variables in $\mathcal{X}^2$ (resp. x_q). Moreover, D is the smallest set that contains all deduction facts $X_q \vdash^? x_q$.

We may note that the set of skeletons actually contains one element up to renaming of variables.

Example 10. Let $t = \mathsf{sdec}(\mathsf{senc}(y,z),z)$. Then we have that

- $\mathsf{Skel}(t,1) = \mathsf{Skel}(t,2) = (\mathsf{sdec}(X_0,X_1), \mathsf{sdec}(x_0,x_1), \{X_0 \vdash^? x_0, X_1 \vdash^? x_1\})$
- $\mathsf{Skel}(t,1\cdot 1) = (\mathsf{sdec}(\mathsf{senc}(X_0,X_1),X_2), \mathsf{sdec}(\mathsf{senc}(x_0,x_1),x_2), \{X_0 \vdash^? x_0, X_1 \vdash^? x_1, X_2 \vdash^? x_2\})$

For a skeleton $(\xi, t, D) \in \mathsf{Skel}(\ell, p)$, the recipe ξ represents the recipe that the attacker will apply on $\xi_0 \vdash^? u_0$ at the position p, the term t represent the corresponding generic term on which the rewrite rule will be applied and finally D is the set of deduction that links the second-order variables with the first-order variables. However, the behavior of a destructor function symbol can be describe by multiple rewrite rules. Hence the application of ξ on $\xi_0 \vdash^? u_0$ may trigger the rewrite rule $\ell \to r$ in one constraint system but may trigger another rewrite rule $\ell' \to r'$ in another constraint system with typically $root(\ell) = root(\ell')$. Hence, since it is impossible to know in advance which rewrite rule will be applied in all constraint systems, we consider all possible cases. The next definition describes the set of generic formulas corresponding to the application of each rewrite rules.

Definition 27. *Let $\xi \in \mathcal{T}(\mathcal{F}, \mathcal{X}^2)$. Let $\ell \to r \in \mathcal{R}$. Let p a position ℓ different from ε. We define the set* $\mathsf{RewF}(\xi, \ell \to r, p) = \{\forall S. \xi \vdash^? r' \Leftarrow D \wedge \ell' =^? t \mid S = vars(\xi, r') \wedge \ell' \to r' \in \mathcal{R}, (\xi, t, D) = \mathsf{Skel}(\ell, p)\}$.

The rule Rew can now be formally defined as follows.

$$\begin{array}{c} \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n\} \\ \to \\ \mathcal{S} \cup \left\{ \begin{array}{l} \{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e{:}\Sigma[\mathsf{F} \to \mathsf{F} \cup \{\psi{:}(\Sigma_0\Sigma\Sigma_1, \mathcal{C}_i^e{:}\Sigma) \mid \psi \in \mathsf{RewF}(\xi, \ell \to r, p)\}])\}_{i=1}^n, \\ \{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{E}^2 \wedge \neg\Sigma])\}_{i=1}^n \end{array} \right\} \end{array} \tag{23}$$

if $\ell \to r \in \mathcal{R}$, $\exists p$ position of ℓ, $\xi \in \mathcal{T}(\mathcal{F}, \mathcal{X}^2_{:k})$ with $k = |\Phi(\mathcal{C}^e_1)|$, $\exists i.\ \exists \psi_0 \in \mathsf{RewF}(\xi, \ell \to r, p)$. $\exists(\xi_0 \vdash^? u_0) \in \mathsf{K}(\mathcal{C}^e_i)$ such that:

- $\Sigma_0 = \{\xi|_p \to \xi_0\}$, $\Sigma \in mgs(\mathcal{C}^e_i|^{\psi_0:(\Sigma_0,\mathcal{C}^e_i)})$; and
- $\psi_0:(\Sigma_0\Sigma, \mathcal{C}^e_i:\Sigma)$ is a solved formula $\forall S.\zeta \vdash^? v \Leftarrow \varphi$ such that Σ_1 is a bijection from $vars^2(\varphi)$ to $\mathcal{N}_{\mathsf{pub}}$; and $\psi_0:(\Sigma_0\Sigma\Sigma_2, \mathcal{C}^e_i:\Sigma) \notin \mathsf{F}_i$ for all bijections from $vars^2(\varphi)$ to $\mathcal{N}_{\mathsf{pub}}$.

13.3 Rule EQ

The third case distinction rule focuses on the static equivalence between solutions of extended constraint systems. More specifically, the rule EQUALITY checks whether a deduction fact $\xi_1 \vdash^? u_1$ from the set of deduction facts K can deduce the same term as another recipe consequence of K. The rule is formalized as follows.

$$\begin{array}{c} \mathcal{S} \cup \{\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}^e_i)\}_{i=1}^n\} \\ \to \\ \mathcal{S} \cup \left\{ \begin{array}{l} \{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}^e_i:\Sigma[\mathsf{F} \to \mathsf{F} \cup \{\psi:(\Sigma_0\Sigma, \mathcal{C}_i:\Sigma)\}])\}_{i=1}^n, \\ \{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}^e_i[\mathsf{E}^2 \wedge \neg\Sigma])\}_{i=1}^n \end{array} \right\} \end{array} \tag{24}$$

if $\exists i.\ \Sigma \in mgs(\mathcal{C}^e_i|^{\psi:(\Sigma_0,\mathcal{C}^e_i)})$, $\psi:(\Sigma_0\Sigma, \mathcal{C}^e_i:\Sigma)$ is solved and:

- either $\Sigma_0 = \{X \to \xi_1, Y \to \xi_2\}$ for some $(\xi_1 \vdash^? u_1), (\xi_2 \vdash^? u_2) \in \mathsf{K}(\mathcal{C}^e_i)$ and for all $(\forall S.H \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}^e_i)$, $H \neq (\xi_1 =^?_f \xi_2)$.
- $\Sigma_0 = \{X \to \xi_1, Y \to \mathsf{f}(X_1, \ldots, X_n)\}$ for some $(\xi_1 \vdash^? u_1) \in \mathsf{K}(\mathcal{C}^e_i)$ and $\mathsf{f}/n \in \mathcal{F}_c$ with $X_1{:}k, \ldots, X_n{:}k$ fresh and for all $(\forall S.\zeta_1 =^?_f \zeta_2 \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}^e_i)$, $\zeta_1 = \xi_1$ implies $root(\zeta_2) \neq \mathsf{f}$.

where $k = |\Phi(\mathcal{C}^e_1)|$ and $\psi = \forall\{X, Y, z\}.(X =^?_f Y) \Leftarrow X \vdash^? z \wedge Y \vdash^? z$ with $X{:}k, Y{:}k, z$ fresh variables.

Note that we consider a generic equality formula $\psi = \forall\{X, Y, z\}.(X =^?_f Y) \Leftarrow X \vdash^? z \wedge Y \vdash^? z$ modeling that X and Y must deduce the same term. Since a recipe consequence of K can either be coming from an deduction fact in K or be a recipe headed by a constructor function symbol, we consider all cases. That is, in ψ, we instantiate X by ξ_1 and we instantiate Y by either ξ_2 where $\xi_2 \vdash^? u_2$ is another deduction facts in K or by $\mathsf{f}(X_1, \ldots, X_n)$ where f is a constructor function symbol and $X_1, \ldots, X_n$ are fresh. Once again, the conditions "for all $(\forall S.H \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}^e_i)$, $H \neq (\xi_1 =^?_f \xi_2)$" and "for all $(\forall S.\zeta_1 =^?_f \zeta_2 \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}^e_i)$, $\zeta_1 = \xi_1$ implies $root(\zeta_2) \neq \mathsf{f}$" are for termination purpose and ensure that we only consider a minimum number of equality formula.

14 Termination and size

We need to be able to bound the size of the most general solutions of the constraint systems we obtain in the partition tree. In this section, we will describe and explain the measure on sets of extended symbolic processes that we use to prove termination of the algorithm and bound the size of most general solutions. More specifically, this section is dedicated to the proof of Theorem 5. To avoid too many notation, we assume in this section that $|\mathcal{R}|_{\mathsf{dag}}$ includes the arity

14.1 Bounding the size of most general solutions

In Theorem 5, we aim to bound the size of the most general unifier of the nodes n in the partition tree, $|\Sigma|_{\mathsf{dag}}$ with $\Sigma \in \mathit{mgs}(n)$. As explained in Section 8, we build the partition tree from the root to leaves. For each node, we build it children by first apply the extended symbolic transition rules in 12 and second by applying as long as we can the simplification and case distinction rules. When no more of the latter rules are applicable, the resulting set of extended symbolic processes formed the children of the node. This sequence of operation is repeated until no more extended symbolic transition are applicable.

To bound the size of the most general solutions, we rely on the fact that when no more simplification and case distinction rules are applicable on a set of symbolic processes S, the most general solution of an extended constraint system $\mathcal{C}^e$ in S is in fact $\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)$ (This property is proven in Appendix). Therefore, to bound the size of the most general solutions of the node of the partition tree, it suffices to bound $|\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)|_{\mathsf{dag}}$, i.e. $|\mathit{st}(\mathit{img}(\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)))|$.

In order to show this bound, we need to introduce some notations. Given an extended constraint system $\mathcal{C}^e = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$, we will denote:

- $\mu^i(\mathcal{C}^e) = \mathit{mgu}(\mathsf{E}^i|_=)$ for $i \in \{1, 2\}$
- $\mathit{PT}(\mathcal{C}^e) = \mathit{st}(\Phi\mu^1(\mathcal{C}^e)) \cup \mathit{st}^1(\mathsf{K}\mu^1(\mathcal{C}^e)) \cup \mathit{st}^1(\mathsf{D}\mu^1(\mathcal{C}^e)) \cup \mathit{st}(\mathit{img}(\mu^1(\mathcal{C}^e))$
- $R(\mathcal{C}^e) = \mathit{st}(\mathit{img}(\mu^2(\mathcal{C}^e))) \cup \mathit{vars}^2(\mathsf{D}) \cup \mathit{st}^2(\mathsf{K}\mu^2(\mathcal{C}^e))$
- $\mathit{st}_c(\mathcal{C}^e) = \mathit{st}_c(\mathit{img}(\mu^2(\mathcal{C}^e)), \mathsf{K}\mu^2(\mathcal{C}^e) \cup \mathsf{D}) \cup \mathit{vars}^2(\mathsf{D})$

Intuitively, $\mu^i(\mathcal{C}^e)$ are the most general solutions of all the equalities in the constraint systems, $\mathit{PT}(\mathcal{C}^e)$ is the set of protocol subterms in the constraint system, $R(\mathcal{C}^e)$ is the set of recipe subterms in the constraint system, and finally $\mathit{st}_c(\mathcal{C}^e)$ represents the consequential subterms of the constraint system. We will write $\mu^1, \mu^2, \mathit{PT}, \ldots$ instead of $\mu^1(\mathcal{C}^e), \mu^2(\mathcal{C}^e), \mathit{PT}(\mathcal{C}^e)$ when the extended constraint system is clear from the context.

Note that $R(\mathcal{C}^e)$ includes $\mathit{st}(\mathit{img}(\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)))$. Therefore, we directly obtain that $|R(\mathcal{C}^e)| \geq |\mathit{st}(\mathit{img}(\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)))|$, i.e. $|R(\mathcal{C}^e)| \geq |\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)|_{\mathsf{dag}}$. Thus, instead of directly bounding $|\mathit{mgu}(\mathsf{E}^2(\mathcal{C}^e)|_=)|_{\mathsf{dag}}$, we will bound $|R(\mathcal{C}^e)|$. We focus on the latter as it is easier to describe its evolution by application of the different rules.

14.1.1 Bounding the size of $R(\mathcal{C}^e)$ by application of a rule.

Let us now consider each rule and determine how $R(\mathcal{C}^e)$ evolves.

- Symbolic rules: Only the rules E-IN and E-OUT increases the size of $R(\mathcal{C}^e)$ by adding at most two new second order variables.
- Simplification rules: Only the rule 20 may increase the size of $R(\mathcal{C}^e)$. Indeed, in such a rule, we add a deduction fact from F in K for each extended constraint systems in the set.
- Case distinction rules: Every case distinction rules split one set of extended symbolic processes in two. One of this set (that we will call the *positive set*) is obtained by applying a most general unifier on each extended constraint systems in the set. The

other set (that we will call the *negative set*) is obtained by adding recipe disequations in all extended constraint systems on the set. Note that in the negative set, $R(\mathcal{C}^e)$ remains unchanged for all extended constraint systems in this set. Therefore, we will focus on the positive set.

In order to bound the size of $R(\mathcal{C}^e)$, we will show that:

P1: $R(\mathcal{C}^e_\emptyset) = \emptyset$ (trivial since the extended constraint is empty)

P2: We can bound the increase of $|R(\mathcal{C}^e)|$ by the rule 20 and by the case distinction rules in the positive set by a polynomial in $|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}} + |\mathcal{R}|_{\mathsf{dag}} + |\mathcal{F}|$.

P3: An extended symbolic process in the partition tree is obtained by at most N applications of the rule 20 and case distinction rules in the positive set where N is an exponential in $|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}} + |\mathcal{R}|_{\mathsf{dag}} + |\mathcal{F}|$.

With this three properties, we can trivially derived that for all extended constraint systems $\mathcal{C}^e$ in all extended symbolic processes obtained during the procedure, $|R(\mathcal{C}^e)|$ is bound by an exponential in $|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}} + |\mathcal{R}|_{\mathsf{dag}} + |\mathcal{F}|$, which conclude the proof of Theorem 5.

The remaining of this section is dedicated to the proof of the second and third properties.

14.1.2 Bounding the increase of $R(C^e)$

When rules 8 to 10 are applied on a constraint system $\mathcal{C}^e$ they do not introduce new variables and decrease the number of symbols in $\mathsf{D}(\mathcal{C}^e)$. These two observations would be enough to bound the size of all $\Sigma \in \mathit{mgs}(\mathcal{C}^e)$. However, our case distinction rules apply these substitutions on other constraint systems which may introduce new variables. Therefore, a measure based on the number of variables is insufficient for sets of extended symbolic processes and we rather use the protocol subterms not already deduced by a recipe in E^2, K and D:

$$\mathcal{M}(\mathcal{C}^e) = \{t \in PT \mid t \notin \mathcal{X}^1 \wedge \forall \xi \in st_c(\mathcal{C}^e) \setminus \mathcal{X}^2.(\xi, t) \notin \textsc{Conseq}(\mathsf{K}\mu^1 \cup \mathsf{D}\mu^1)\}$$

Consider first the simplification rule 11 and the ones from Figure 11. They typically apply protocol term substitutions on the constraint system (they also effect recipe disequations that are irrelevant in $\mathcal{M}(\mathcal{C}^e)$). Note that the applied substitution is always generated from terms already in the constraint system. As such $\mu^1(\mathcal{C}^e\!\downarrow) = \mu^1(\mathcal{C}^e)$ and so $\Phi(\mathcal{C}^e\!\downarrow)\mu^1(\mathcal{C}^e\!\downarrow) = \Phi(\mathcal{C}^e)\mu^1(\mathcal{C}^e)$, $\mathsf{K}(\mathcal{C}^e\!\downarrow)\mu^1(\mathcal{C}^e\!\downarrow) = \mathsf{K}(\mathcal{C}^e)\mu^1(\mathcal{C}^e)$ and $\mathsf{D}(\mathcal{C}^e\!\downarrow)\mu^1(\mathcal{C}^e\!\downarrow) = \mathsf{D}(\mathcal{C}^e)\mu^1(\mathcal{C}^e)$. Thus, we directly obtain that $|\mathcal{M}(\mathcal{C}^e\!\downarrow)| \leq |\mathcal{M}(\mathcal{C}^e)|$.

Let us now look at the rules 8, 9 and 10 and let us consider $\mathcal{C}^e \xrightarrow{\Sigma} \mathcal{C}'^e$. The rule 8 does not modify the protocol terms of the constraint systems by apply a recipe substitution. However, we show an invariant on the constraint systems that any $\xi, \zeta \in st_c(\mathcal{C}^e)$ are consequence of $\mathsf{K} \cup \mathsf{D}$ as well as any of their subterms (see Definition 34 in Appendix). Thus, we deduce from the definition of $st_c(\mathcal{C}^e)$ that $st_c(\mathcal{C}^e)\Sigma \subseteq st_c(\mathcal{C}'^e)$. To conclude that $|\mathcal{M}(\mathcal{C}'^e)| \leq |\mathcal{M}(\mathcal{C}^e)|$, we rely on the following lemma:

Lemma 14. *Let S, S' be two sets of solved deduction facts. Let $\varphi = \{X_i \vdash^? u_i\}_{i=1}^n$ such that all X_i are pairwise distinct. For all Σ, σ, for all $(\xi, t) \in \textsc{Conseq}(\mathsf{S} \cup \varphi)$, if for all $i \in \{1, \ldots, n\}$, $(X_i\Sigma, u_i\sigma) \in \textsc{Conseq}(\mathsf{S}\Sigma\sigma \cup \mathsf{S}')$ then $(\xi\Sigma, t\sigma) \in \textsc{Conseq}(\mathsf{S}\Sigma\sigma \cup \mathsf{S}')$.*

Typically, this lemma (proven in Appendix) indicates that the notion of consequence is stable by application of recipe and protocol substitutions (under some conditions). In other words, if $(\xi, t) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}^e)\mu^1(\mathcal{C}^e) \cup \mathsf{D}(\mathcal{C}^e)\mu^1(\mathcal{C}^e))$ then $(\xi\Sigma, t) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}'^e)\mu^1(\mathcal{C}'^e) \cup \mathsf{D}(\mathcal{C}'^e)\mu^1(\mathcal{C}'^e))$. Since $st_c(\mathcal{C}^e)\Sigma \subseteq st_c(\mathcal{C}'^e)$, we conclude that $|\mathcal{M}(\mathcal{C}'^e)| \leq |\mathcal{M}(\mathcal{C}^e)|$.

By applying the same reasonning for the rule 10, we can also show that $\mathcal{M}(C'^e) \leq \mathcal{M}(\mathcal{C}^e)$. However, we can even show that this inequality is strict. Indeed, using the same the notation in the rule 10, this rule is only applied if $\mathcal{C}^e = \mathcal{C}^e\!\downarrow$ and for all $\xi \in st_c(\mathcal{C}^e) \setminus \{X\}$, $(\xi, u) \notin \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$. Note that $\mathcal{C}^e = \mathcal{C}^e\!\downarrow$ implies that $\mathsf{K}\mu^1 = \mathsf{K}$ and $\mathsf{D}\mu^1 = \mathsf{D}$. Moreover, it also implies that $u \in \mathcal{M}\mathcal{C}^e$. However, in $\mathcal{C}'^e$, we have that $(\xi, u\mu_1(\mathcal{C}'^e)) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}'^e)\mu^1(\mathcal{C}'^e) \cup \mathsf{D}(\mathcal{C}'^e)\mu^1(\mathcal{C}'^e))$. Moreover, we show another invariant on the constraint system (see Definition 34 in Appendix) that ensures us that $X \in st_c(\mathcal{C}^e)$ and so $\xi \in st_c(\mathcal{C}'^e)$. Hence, we obtain that $u\mu^1(\mathcal{C}'^e) \notin \mathcal{M}(\mathcal{C}'^e)$ allowing us to conclude that $\mathcal{M}(C'^e) < \mathcal{M}(C^e)$. By applying the same reasoning, we can also show that $\mathcal{M}(C'^e) < \mathcal{M}(C^e)$ when the rule 9 is applied.

Let us now look at the evolution of $R(\mathcal{C}^e)$ when applying all these rules: The simplification rules from Figure 11 only affects the recipe disequations which does not influence $R(\mathcal{C}^e)$; the rule 11 does not modify the recipes of the constraint system. Rules 8 and 10 apply a recipe substitution Σ on the constraint system where $\Sigma = mgu(\xi, \zeta)$ with $\xi, \zeta \in R(\mathcal{C}^e)$. Thus, $|R(\mathcal{C}'^e)| \leq |R(\mathcal{C}^e)|$. Finally, the only rule that increases $R(\mathcal{C}^e)$ is the rule 9 since it generates n fresh variables $X_1, \ldots, X_n$ where n is the arity of the symbol f.

Since we proved that $|\mathcal{M}(C'^e)| < |\mathcal{M}(C^e)|$ when the rule 9 is applied and $|\mathcal{M}(C^e)|$ never increase for any other rules, we conclude that:

$$\text{for all } \Sigma \in mgs(\mathcal{C}^e), |R(\mathcal{C}^e{:}\Sigma)| \leq |R(\mathcal{C}^e)| + |\mathcal{F}| \times (|\mathcal{M}(\mathcal{C}^e)| - |\mathcal{M}(\mathcal{C}^e{:}\Sigma)|) \tag{25}$$

Note that as we are only interested of the complexity class of the problem, we will over approximate the bounds. As such, we over approximate the maximal arity of the signature by $|\mathcal{F}|$.

Recall that the case distinction rule Sat always a most general solution of one of the extended constraint systems to all the extended symbolic processes in the positive set. Thus, the equation 25 describes how $R(\mathcal{C}^e)$ increases for each application of the rule Sat. By bounding $|\mathcal{M}(\mathcal{C}^e)|$ by a polynomial in $|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}} + |\mathcal{R}|_{\mathsf{dag}} + |\mathcal{F}|$, we would prove the property P2 for the rule Sat.

Evolution and bound of $|\mathcal{M}(\mathcal{C}^e)|$. We showed that for all $\Sigma \in mgs(\mathcal{C}^e)$, $|\mathcal{M}(\mathcal{C}^e{:}\Sigma)| \leq |\mathcal{M}(\mathcal{C}^e)|$. In fact we can generalize this property to a more general set of substitution Σ:

Definition 28. *Let $\mathcal{C}^e$ be an extended constraint system. Let Σ be a second-order substitution. We say that $\Sigma \in \textsc{CompSub}(\mathcal{C}^e)$ if $dom(\Sigma) \subseteq vars^2(\mathsf{D}(\mathcal{C}^e))$ and for all $X \in dom(\Sigma)$, $X\Sigma \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}^e) \cup \mathsf{D}' \cup D_\Sigma)$ where $\mathsf{D}' = \{X \vdash^? u \in \mathsf{D}(\mathcal{C}^e) \mid X \notin dom(\Sigma)\}$ and $D_\Sigma = \{X \vdash^? x \mid x$ fresh and $X \in vars^2(\Sigma) \setminus vars^2(\mathcal{C}^e)\}$.*

Intuitively, $\textsc{CompSub}(\mathcal{C}^e)$ represents the recipe substitutions Σ that can be applied be applied to the constraint system $\mathcal{C}^e$, i.e. $\mathcal{C}^e{:}\Sigma$, and such that the recipes in the of Σ would be consequence of $\mathcal{C}^e{:}\Sigma$. Note that $mgs(\mathcal{C}^e) \subseteq \textsc{CompSub}(\mathcal{C}^e)$.

By applying Lemma 14, we can show that:

$$\text{for all } \Sigma \in \textsc{CompSub}(\mathcal{C}^e), |\mathcal{M}(\mathcal{C}^e{:}\Sigma)| \leq |\mathcal{M}(\mathcal{C}^e)| \tag{26}$$

Note that in a set of symbolic processes two extended constraint systems $\mathcal{C}_1^e, \mathcal{C}_2^e$ always have the same *recipe structure*, i.e. $|\Phi(\mathcal{C}_1^e)| = |\Phi(\mathcal{C}_2^e)|$, $\mathit{vars}^2(\mathcal{C}_1^e) = \mathit{vars}^2(\mathcal{C}_2^e)$ and $\{\xi \mid (\xi \vdash^? u) \in \mathsf{K}(\mathcal{C}_1^e)\} = \{\xi \mid (\xi \vdash^? u) \in \mathsf{K}(\mathcal{C}_2^e)\}$. Thus, we deduce that $\textsc{CompSub}(\mathcal{C}_1^e) = \textsc{CompSub}(\mathcal{C}_2^e)$. Therefore, we can conclude that for any simplification and case distinction rules, $|\mathcal{M}(\mathcal{C}^e)|$ never increase for all extended constraint systems in a set of extended symbolic processes.

Let us now bound $|\mathcal{M}(\mathcal{C}^e)|$. In order to do so we explore the relation between $\mathcal{C}$ and $\mathcal{C}^e$ in an extended symbolic process $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e)$. Recall that in all rules of the extended symbolic semantics (figure 12), we always add the same equations, disequations or deduction fact in $\mathcal{C}$ and $\mathcal{C}^e$. Thus, if we see $\mathcal{C}$ as an extended constraint system with $\mathsf{K}(\mathcal{C}) = \mathsf{F}(\mathcal{C}) = \emptyset$ and $\mathsf{E}^2(\mathcal{C}) = \emptyset$ then we obtain that $\mathcal{M}(\mathcal{C}^e)$ increases at most as much as $\mathcal{M}(\mathcal{C})$ when applying a symbolic extended rules. Combining with the fact that $\mathcal{C}$ is always left untouched by simplification and case distinction rules, and that $\mathcal{M}(\mathcal{C}^e)$ never increase by these rules, we obtain:

$$\text{for all } (\mathcal{P}, \mathcal{C}, \mathcal{C}^e) \text{ in a set of extended symbolic processes}, |\mathcal{M}(\mathcal{C}^e)| \leq |\mathcal{M}(\mathcal{C})| \qquad (27)$$

This relation is very useful as $|\mathcal{M}(\mathcal{C})|$ can easily be bound by a polynomial in $|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}} + |\mathcal{R}|_{\mathsf{dag}}$. Indeed, each symbolic rule add protocol terms from the process P and Q, and in the case of the rule E-IN and E-OUT also at at most 1 fresh first order variables. This property is formalise in this simple lemma:

Lemma 15. *For all $(\{\!\{P\}\!\}, \mathcal{C}_\emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$, there exists $S = \{t_i =^? u_i\}_{i=1}^n$ such that:*

- *for all $i \in \{1, \dots, n\}$, $t_i, u_i \in \mathit{st}(P) \cup \{y_i\}_{i=1}^{|\mathsf{tr}|}$*
- *$\mu^1(\mathcal{C}') \in \mathit{mgu}_{\mathcal{R}}(S)$*

Proof. This lemma is proved by induction on the number of transition in $(\{\!\{P\}\!\}, \mathcal{C}_\emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{s}} (\mathcal{P}', \mathcal{C}')$ and is direct case distinction on the symbolic rule applied. □

Moreover, by adapting the classical algorithm for computing most general unifier, we can show that for all $\sigma \in \mathit{mgu}_{\mathcal{R}}(S)$, $|\mathit{st}(\sigma)| \leq |\mathit{st}(S)| + |\mathcal{R}|_{\mathsf{dag}} \times |\{t \in \mathit{st}(S) \mid \mathit{root}(t) \in \mathcal{F}_d\}|$. By combining this property with lemma 15 and with the fact that an extended symbolic process is obtained by at most $\max(|P|_{\mathsf{dag}}, |Q|_{\mathsf{dag}})$ application of symbolic rules, we conclude that for all extended symbolic processes $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e)$,

$$|\mathcal{M}(\mathcal{C}^e)| \leq |\mathcal{M}(\mathcal{C})| \leq 2(|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}}) \times (|\mathcal{R}|_{\mathsf{dag}} + 1) \qquad (28)$$

Combining this equation with equation 25, we obtain that:

$$\text{for all } \Sigma \in \mathit{mgs}(\mathcal{C}^e), |R(\mathcal{C}^e{:}\Sigma| \leq |R(\mathcal{C}^e)| + 2(|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}}) \times |\mathcal{F}| \times (|\mathcal{R}|_{\mathsf{dag}} + 1) \qquad (29)$$

14.1.3 Most general solutions and formulas

Equation 29 allows us to bound the increase of $|R(\mathcal{C}^e)|$ every time the rule SAT is applied in the case (a) (see Section 13.1). However for the other cases and the rules REW and EQUALITY, we compute and apply most general unifier of an extended constraint system combined with a deduction or equality formula, i.e. $\Sigma \in \mathit{mgs}(\mathcal{C}^e|^{\psi})$ for some formula ψ, or with a protocol disequation.

Let us focus on the case of the rule REW (the case of the rule EQUALITY follows the same logic). Note that equation 25 can be instantiated with $\mathcal{C}^e|^{\psi}$ as follows:

$$\text{for all } \Sigma \in mgs(\mathcal{C}^e|^{\psi}), |R(\mathcal{C}^e|^{\psi}{:}\Sigma)| \leq |R(\mathcal{C}^e|^{\psi})| + |\mathcal{F}| \times (|\mathcal{M}(\mathcal{C}^e|^{\psi})| - |\mathcal{M}(\mathcal{C}^e|^{\psi}{:}\Sigma)|) \quad (30)$$

However in the rule REW, the formula ψ is in fact $\Sigma_0{:}(\psi_0, \mathcal{C}^e_i)$ with $\psi_0 \in \mathsf{RewF}(\xi, \ell \to r, p)$ and $\ell \to r \in \mathcal{R}$. By definition of $\mathsf{RewF}(\xi, \ell \to r, p)$, we obtain that $|st^2(\psi_0)| \leq |\mathcal{R}|_{\mathsf{dag}} \times |\mathcal{F}|$ and $|st^1(\psi_0)| \leq |\mathcal{R}|_{\mathsf{dag}}(1+|\mathcal{F}|)$. Hence, $|\mathcal{M}(\mathcal{C}^e_i|^{\psi_0:(\Sigma_0,\mathcal{C}^e_i)})| \leq |\mathcal{M}(\mathcal{C}^e_i)|+|\mathcal{R}|_{\mathsf{dag}}(1+|\mathcal{F}|)$ and $|R(\mathcal{C}^e_i) \cup st^2(\psi_0{:}(\Sigma_0, \mathcal{C}^e_i))| \leq |R(\mathcal{C}^e_i)|+|\mathcal{R}|_{\mathsf{dag}} \times |\mathcal{F}|$. Finally, notice that $|R(\mathcal{C}^e_i{:}\Sigma)| \leq |R(\mathcal{C}^e_i|^{\psi_0:(\Sigma_0,\mathcal{C}^e_i)}{:}\Sigma)|$. Thus we deduce the following statement:

$$\text{for all } \Sigma \in mgs(\mathcal{C}^e|^{\Sigma_0:(\psi_0,\mathcal{C}^e_i)}), |R(\mathcal{C}^e_i{:}\Sigma)| \leq |R(\mathcal{C}^e_i)| + |\mathcal{F}||\mathcal{M}(\mathcal{C}^e)| + |\mathcal{R}|_{\mathsf{dag}}|\mathcal{F}|(2 + |\mathcal{F}|) \quad (31)$$

Since we already bound $|\mathcal{M}(\mathcal{C}^e)|$ by $2(|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}}) \times (|\mathcal{R}|_{\mathsf{dag}} + 1)$, we conclude once again that the increase of $R(\mathcal{C}^e)$ is bound by a polynomial.

Finally, note that once a formula is created and added into F, the only modification on its recipes is the application of substitution during the application of other rules. However, in such a case, we can easily show the following result:

Lemma 16. *Let $\mathcal{C}^e$ be an extended constraint system. Let $\psi \in \mathsf{F}(\mathcal{C}^e)$. Let $\Sigma \in$ COMPSUB$(\mathcal{C}^e)$. For all $k \in \mathbb{N}$, it $|R(\mathcal{C}^e) \cup st^2(\psi)| \leq |R(Ce)| + k$ then $|R(\mathcal{C}^e{:}\Sigma) \cup st^2(\psi\Sigma)| \leq |R(\mathcal{C}^e{:}\Sigma)| + k$.*

Intuitively, this lemma allows us to show that whenever the simplification rule 20 is applied, i.e., whenever a deduction fact $\psi \in \mathsf{F}(\mathcal{C}^e)$ is added to $\mathsf{K}(\mathcal{C}^e)$, the increase of $R(\mathcal{C}^e)$ can directly characterized at the time ψ was first created. This allows us to deduce $|R(\mathcal{C}^e) \cup st^2(\psi)| \leq |R(\mathcal{C}^e)| + |\mathcal{R}|_{\mathsf{dag}}|\mathcal{F}|$ and to conclude the proof of Property P2.

14.2 Bounding the number of rules

Each case distinction and simplification rule replaces a set of extended symbolic processes by one or two new sets of extended symbolic processes, leaving all the other sets unchanged. Thus, in a derivation $\mathcal{S} \to^* \mathcal{S}'$ and given a set $S \in \mathcal{S}'$, we are in fact interested to measure the number of applications of rules that were specifically needed to obtain S. To do so, we consider a measure on sets of extended symbolic processes $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}^e_i)\}_{i=1}^n$ and we show that this measure strictly decreases by application of any rules (including the symbolic transition rules). This measure is in fact a 10-tuple of integers where each element focuses on a specific part of the set of extended symbolic processes and we consider the lexicographic order on tuples. In the rest of this section, we will describe each element of this measure and how to bound them.

First element: size of the process. As first element of the measure, we compute a maximum on the sizes of the processes in the multisets $\mathcal{P}_i$, *i.e.* $\max(\sum_{R \in \mathcal{P}_i} |R|_{\mathsf{dag}} \mid i = 1 \ldots n)$. Notice that this stays unchanged for any simplification or case distinction rules but strictly decreases when applying the extended symbolic transitions. Moreover, since the size of a process consider the explicit tree structure, we have that this first element of the tuple is bounded by the size of the initial processes P and Q.

Second element: Number of constraint systems. The second element of the measure considers the number of extended symbolic processes in the set, *i.e.* n, that may increase only when applying a symbolic transition but these rules are already covered by the first element of the measure due to the lexicographic order on tuples. Note that n strictly decreases when applying the simplification rules 18 and 19. Moreover, it also strictly decrease when the rule Sat is applied with the case (c) of its application conditions. In such a case, we consider a disequations ψ and $\Sigma \in mgs(\mathcal{C}^e_j|^{\psi})$. Thus, Σ represents the solutions of $\mathcal{C}^e_j$ that will not satisfy ψ. Hence, in the symbolic process $(\mathcal{P}_j, \mathcal{C}_j, \Sigma{:}\mathcal{C}^e_j)$, the extended constraint system $\Sigma{:}\mathcal{C}^e_j$ will be reduced to $\bot$ by the simplification rules, leading to an application of rule 18.

Intuitively, n corresponds to the number of symbolic transitions possible from P and Q for a given symbolic trace. Hence, we can bound n by $(|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}}$. Notice that the part $|\mathcal{R}|_{dag}^{|P|_{dag}}$ is due to the computation of the most general unifiers modulo $\mathcal{R}$ in the symbolic transitions. Moreover, notice that this bound is exponential in the size of the inputs P, Q and $\mathcal{R}$.

Third element: Number of terms not consequence. Given $\mathcal{C}$ an extended symbolic constraint system, let us denote $\mathsf{set}_\mathsf{K}(\mathcal{C})$ the following set:

$$\{t \in st^1(\Phi(\mathcal{C})) \cup st^1(\mathsf{D}(\mathcal{C})) \mid t \not\in \textsc{Conseq}(\mathsf{K}(\mathcal{C}) \cup \mathsf{D}(\mathcal{C}))\}$$

In an extended symbolic process $(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}^e_i)$ with $i \in \{1, \dots, n\}$, the set $\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)$ represents the terms in the constraint systems that are not consequence of $\mathsf{K}(\mathcal{C}^e_i)$ and $\mathsf{D}(\mathcal{C}^e_i)$. Typically, it corresponds to the terms that are not deducible by the attacker but could potentially be. In fact, we can show that when the simplification rule 20 is applied, i.e. when a deduction fact is added to $\mathsf{K}(\mathcal{C}^e_i)$, the head protocol term of this deduction fact is necessarily a subterm of the frame (thanks to the fact that we only consider subterm rewriting systems). Moreover, since by the application conditions of the simplification rule, we have that the head protocol term is not already consequence, the size of $\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)$ will strictly decrease. Finally, since all case distinction rules consist of apply substitutions Σ whose sub-recipe are all consequence (since they are in fact most general solutions of the constraint systems), $|\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)|$ never increase by application of any case distinction rules.

Finally, with the same arguments for showing $\mathcal{M}(\mathcal{C}^e_i) \leq \mathcal{M}(\mathcal{C}_i)$, we can show that $|\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)|$ is bounded by $\mathcal{M}(\mathcal{C}_i)$ and so $|\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)| \leq 2(|P|_\mathsf{dag} + |Q|_\mathsf{dag}) \times (|\mathcal{R}|_\mathsf{dag} + 1)$. Therefore, as third element, we consider $\min(|\mathsf{set}_\mathsf{K}(\mathcal{C}^e_i)| \mid i = 1 \dots n)$ which is bounded by $2(|P|_\mathsf{dag} + |Q|_\mathsf{dag}) \times (|\mathcal{R}|_\mathsf{dag} + 1)$.

Fourth element: Number of unsolved extended constraint systems. An extended constraint system $\mathcal{C}$ is unsolved when the deduction facts in $\mathsf{D}(\mathcal{C})$ do not have pairwise distinct variables as right hand term. However, given $\Sigma \in mgs(\mathcal{C})$, the application of Σ on $\mathcal{C}$, i.e. $\Sigma{:}\mathcal{C}$, gives a solved extended constraint system (this property can be obtained in fact directly from the definition of most general solutions). Therefore, when applying the rule Sat with the case (a) of its application conditions, we directly obtain that the number of unsolved extended constraint systems strictly decrease.

Note that we trivially deduce that the number of unsolved systems is bounded by n and so the fourth element is bounded by $(|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}}$.

Fifth element: Application of the rule REW. The fifth element represents the number of application of the rule Rew that is still possible. Typically, we consider all the parameters of the rule Rew (the deduction facts from K, the rewrite rule, etc) on which the rule Rew would be applied with a most general solutions that does not already corresponds to a deduction fact in F. To define this number, we consider the following set $\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)$.

Definition 29. *Let $\mathcal{C}^e$ be an extended symbolic process. We denote $\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)$ the set of tuples $(\psi, \ell \to r, p, \psi_0, \Sigma)$ that satisfies all the application conditions of the rule* Rew.

Obviously, $|\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)|$ strictly decrease when applying the rule Rew. For the case distinction rules, i.e. Sat and Equality, do not increase $|\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)|$. Indeed consider a deduction formula ψ in $\mathsf{F}(\mathcal{C}^e)$ obtained by application of the rule Rew, i.e. corresponding to some parameters $(\psi, \ell \to r, p, \psi_0, \Sigma)$. Now assume that due to the application of the rule Sat or Equality, we apply a substitution Σ' on the constraint system $\mathcal{C}^e$. In such a case, the formula $\psi\Sigma' \in \mathsf{F}(\mathcal{C}^e{:}\Sigma')$ would in fact correspond to the application of the rule Rew with parameters $(\psi\Sigma', \ell \to r, p, \psi_0, \Sigma\Sigma')$. Note that $|\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)|$ may increase by application of rule 20 since $|\mathsf{K}(\mathcal{C}^e)|$ will increase. However, this rule is already covered by the third element of the measure.

Bounding the size of $|\mathsf{set}_{\text{Rew}}(\mathcal{C}^e)|$ can easily be done: The number of $\psi \in \mathsf{K}$ possible is bounded by $|\mathsf{K}|$, itself bounded by $|\mathsf{set}_{\mathsf{K}}(\mathcal{C}^e)|$ (see the explanation of the third element). The number of rewrite rules, position p and $\psi_0 \in \mathsf{RewF}(\xi, \ell \to r, p)$ only depends on the rewrite systems and can be bounded by $|\mathcal{R}|_{\mathsf{dag}}^{|\mathcal{R}|_{\mathsf{dag}}}$. Note that the exponential comes mainly from the number of possible positions in ℓ. We already know that the number of most general solutions is bounded by $(\mathsf{K}(\mathcal{C})| + 1)^{\mathcal{M}(\mathcal{C}^e)}$.

Since rule Rew can be triggered by any extended constraint systems in $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n$, we consider $\sum_{i=1}^n |\mathsf{set}_{\text{Rew}}(\mathcal{C}_i^e)|$ as fifth element of the measure. Combining all the previous results, we obtain the following result:

$$\begin{array}{rcl}\sum_{i=1}^n |\mathsf{set}_{\text{Rew}}(\mathcal{C}_i^e)| & \leq & ((|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}}) \times |\mathcal{R}|_{\mathsf{dag}}^{|\mathcal{R}|_{\mathsf{dag}}} \times \\ & & \big((|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}})(1 + |\mathcal{R}|_{\mathsf{dag}})\big)^{(|P|_{\mathsf{dag}}+|Q|_{\mathsf{dag}})(1+|\mathcal{R}|_{\mathsf{dag}})+1}\end{array}$$

Sixth element: Number of unsolved deduction formulas. We say that a deduction formulas is unsolved when it is not a deduction fact. When we apply the rule Rew, we potentially add multiple unsolved deduction formulas and no other rule can increase the number of deduction formulas. Moreover, when we apply the rule Sat with the case (b) of its application conditions on deduction formulas, we have an unsolved deduction formula $\psi \in \mathsf{F}(\mathcal{C}_j^e)$ and $\Sigma \in mgs(\mathcal{C}_j^e|_{\mathcal{T}_\psi^2})$. Hence, Σ corresponds to the solutions of $\mathcal{C}_j^e$ that trivially satisfy the hypotheses of ψ. Thus when applying Σ on $\mathcal{C}_j^e$, i.e. $\mathcal{C}_j^e{:}\Sigma$, the formula $\psi\Sigma$ will becomes solved. This allows us to deduce that the number of unsolved deduction formulas will strictly decrease by application of the rule Sat with the case (b) of its application conditions.

We therefore consider $|\{\psi \in \mathsf{F}(\mathcal{C}_i^e) \mid i \in \{1, \ldots, n\} \wedge \psi \text{ is an unsolved deduction formula}\}|$ as sixth element of the measure. To bound this number, we need to recall that we always apply the case distinction rules with the following priority order: Sat ¡ Equality ¡ Rew. Thus, when we apply a rule Rew, there is no unsolved deduction formula in any of the extended constraint systems (otherwise we should have applied the rule Sat). It means to the sixth element of the measure is bounded by the number of deduction formula produced by one instance of Rew. By definition, we know that $|\mathsf{RewF}(\xi, \ell \to r, p)| \leq |\mathcal{R}|$ (one formula per rewrite rule). Thus, the rule REW generates at most $|\mathcal{R}| \times n$ deduction formula which is bounded by $|\mathcal{R}|_{\mathsf{dag}} \times \big((|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}}\big)$.

Seventh element: Application of the rule EQ. Similarly to the fifth element that corresponds to the number of application of the rule REW, the seventh element of the measure will considered the maximal number of remaining application of the rule EQUALITY.

The application conditions stipulate that the rule can be applied either (a) on two deduction facts of $\mathsf{K}(\mathcal{C}_i^e)$, or (b) on one deduction fact of $\mathsf{K}(\mathcal{C}_i^e)$ in combination with a construction function symbol.

Note that even though the rule also consider the existence of a most general solution $\Sigma \in mgs(\mathcal{C}_i^e|^{\psi:(\Sigma_0,\mathcal{C}_i^e)})$, the number of application of the rule EQUALITY will not depend on the number of possible most general solutions. Indeed, consider the case (a) where the rule is applied on two deduction fact $(\xi_1 \vdash^? u_1), (\xi_2 \vdash^? u_2) \in \mathsf{K}(\mathcal{C}_i^e)$. Thus, an equality formula with $\xi_1\Sigma =_f^? \xi_2\Sigma$ as head will be added in $\mathsf{F}(\mathcal{C}_i^e{:}\Sigma)$. However, in the application conditions of the rule, we also require that *for all* $(\forall S.H \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}_i^e)$, $H \neq (\xi_1 =_f^? \xi_2)$. Thus, a new application of the rule EQUALITY on $\mathcal{C}_i^e{:}\Sigma$ with the same (up to instantiation of Σ) deductions facts from $\mathsf{K}(\mathcal{C}_i^e{:}\Sigma)$ will be prevented.

The same situation occurs in case (b) with the condition *for all* $(\forall S.\zeta_1 =_f^? \zeta_2 \Leftarrow \varphi) \in \mathsf{F}(\mathcal{C}_i^e)$, $\zeta_1 = \xi_1$ *implies* $root(\zeta_2) \neq \mathsf{f}$.

We therefore conclude that the rule EQUALITY can be applied only once per pair of deduction facts in K and once per deduction fact in K and function symbol in $\mathcal{F}_c$.

Definition 30. *Let $\mathcal{C}^e$ be an extended symbolic process. We denote $\mathsf{set}_{\mathrm{EQ}}(\mathcal{C}^e)$ the set of pairs $(\psi, \psi') \in \mathsf{K}(\mathcal{C}^e)^2$ and pairs $(\psi, \mathsf{f}) \in \mathsf{K} \times \mathcal{F}_c$ that satisfy all the application conditions of the rule* EQUALITY.

We therefore we consider $\sum_{i=1}^n |\mathsf{set}_{\mathrm{EQ}}(\mathcal{C}_i^e)|$ as seventh element of the measure and we obtain the following result:

$$\begin{array}{rcl}\sum_{i=1}^n |\mathsf{set}_{\mathrm{EQ}}(\mathcal{C}_i^e)| & \leq & ((|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}})\times \\ & & 2|\mathcal{R}|_{\mathsf{dag}}(|P|_{\mathsf{dag}} + |Q|_{\mathsf{dag}})^2(1 + |\mathcal{R}|_{\mathsf{dag}})^2\end{array}$$

Eighth element: Number of unsolved equality formulas. Unsolved equality formulas can be generated by two rules: the case distinction rule EQUALITY or the simplification rule 21. However, once again because of the priority order SAT < EQUALITY < REW, the two rules cannot be triggered simultaneously and the rule EQUALITY is only triggered when there is no unsolved equality formulas. Note that due to the condition $\forall i.\forall(\forall S.\zeta_1 =_f^? \zeta_2 \Leftarrow \varphi) \in \mathsf{F}_i$, $\zeta_1 \neq \xi$ *or* $\zeta_2 \neq \xi$ in rule 21, two instances of the rule 21 with different recipes ζ (e.g. if u_1 can be deducible with two different recipes) cannot be applied sequentially. Thus, at any given moment, there is at most one unsolved equality formula per extended constraint system $\mathcal{C}_i^e$, for all $i \in \{1, \dots, n\}$.

We therefore consider $|\{\psi \in \mathsf{F}(\mathcal{C}_i^e) \mid i \in \{1, \dots, n\} \wedge \psi \text{ is an unsolved equality formula}\}|$ as eighth element of the measure which is bounded by n and so by $(|P|_{dag}|\mathcal{R}|_{dag}|)^{|P|_{dag}} + (|Q|_{dag}|\mathcal{R}|_{dag}|)^{|Q|_{dag}}$.

Finally, notice that when we apply the rule SAT with the case (b) of its application conditions on equality formulas, $|\{\psi \in \mathsf{F}(\mathcal{C}_i^e) \mid i \in \{1, \dots, n\} \wedge \psi \text{ is an unsolved equality formula}\}|$ strictly decreases.

Tenth element: Remaining most general solutions. So far, every time we showed that one of the previous element of the measure (strictly) decrease by application of a case distinction rule, we always focused on the set of extended symbolic processes corresponding to the application of the most general solution Σ (e.g. the set $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e{:}\Sigma)\}_{i=1}^n$ for the rule Sat). This is due to the fact that the other set of extended symbolic process generated by the rule only add recipe disequations, i.e. $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{E}^2 \wedge \neg\Sigma])\}_{i=1}$. These recipe disequations never increase any of the previous elements of the measures but strictly decreases the number of most general solutions we can calculate for the same instance of the rule. For example, if $\Sigma \in mgs(\mathcal{C}^e)$ then $|mgs\mathcal{C}^e| > |mgs(\mathcal{C}^e[\mathsf{E}^2 \wedge \neg\Sigma])|$. Hence, by considering as tenth element of the measure the number of most general solutions available for any possible instance of a rule applicable on $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e)\}_{i=1}^n$, we ensure that the measure strictly decrease when considering the set $\{(\mathcal{P}_i, \mathcal{C}_i, \mathcal{C}_i^e[\mathsf{E}^2 \wedge \neg\Sigma])\}_{i=1}$.

Note however that it is unnecessary to bound this tenth element. We only require it to be finite for termination purposes, which is the case. Indeed, recall that we aim to bound the number of time a most general solution is applied in order to bound $|R(\mathcal{C}_i^e)|$. Since this tenth element of the measure focus on the parts of the rules that do not apply most general solution, it does not impact the bound of $|R(\mathcal{C}_i^e)|$.

15 Implementation

Building on the previous section we have implemented a prototype in OCaml, called Deepsec (DEciding Equivalence Properties in SECurity protocols), for verifying trace equivalence. The tool's specification language extends the grammar presented in section 2.2: in particular, we define a non-deterministic choice operator $P + Q$, a let operator for variable assignment $\mathsf{let}\ x = u\ \mathsf{in}\ P\ \mathsf{else}\ Q$, as well as bounded replication $!^n P$ defining n copies of P in parallel. These additional primitives are only here for modelling convinience—and the native integration allowed specific optimisations compared to encoding within the initial calculus. The syntax and structure of Deepsec's input files are similar to the ones of the widely used ProVerif [BSC16] tool. We hope this will make it easier for new users to discover and handle our tool.

Partial order reductions The tool also implements *partial order reductions* (POR), an optimisation technique for protocol analysis developed by Baelde et al. [BDH15]. The basic idea is to discard part of the state space that is redundant. This optimisation is sound when processes are *action determinate*, as defined in [BDH15]. Assigning a different channel name to each parallel process is a simple, syntactic way to ensure this property although this is not always possible—typically when looking at anonymity or unlinkability properties. In practice, Deepsec automatically detects action-determinate processes and activates the POR, which drastically reduces the number of symbolic executions that need be considered.

Distributing the computation The main task of Deepsec is to generate a partition tree. This task can be distributed: computing a given node of the tree can be done independently of its sibling nodes. However, some engineering is needed to avoid heavy communication overhead due to task scheduling. Indeed, the partition tree is indeed not a balanced tree and it is impossible to know which branches will be larger than others. Hence, in practice we

	Protocol (# of roles)		Akiss	APTE	SPEC	Sat-Eq	DeepSec
Strong secrecy	Denning-Sacco	3	✓ <1s	✓ <1s	✓ 11s	✓ <1s	✓ <1s
		6	✓ <1s	✓ 1s	🛑	✓ <1s	✓ <1s
		7	✓ 6s	✓ 3s		✓ <1s	✓ <1s
		10	🛑	✓ 9m49		✓ <1s	✓ <1s
		12		⏰		✓ <1s	✓ <1s
		14				✓ <1s	✓ <1s
		17				✓ <1s	✓ 1s
		29				✓ <1s	✓ 6s
	Wide Mouth Frog	3	✓ <1s	✓ <1s	✓ 5s	✓ <1s	✓ <1s
		6	✓ <1s	✓ <1s	✓ 1h11m	✓ <1s	✓ <1s
		7	✓ <1s	✓ 1s	🛑	✓ <1s	✓ <1s
		10	✓ 10s	✓ 3m35		✓ <1s	✓ 1s
		12	✓ 22m16s	⏰		✓ <1s	✓ <1s
		14	🛑			✓ <1s	✓ <1s
		17				✓ <1s	✓ 1s
		23				✓ <1s	✓ 3s
	Yahalom-Lowe	3	✓ <1s	✓ <1s	✓ 7s	✓ <1s	✓ <1s
		6	✓ 2s	✓ 41s	🛑	✓ <1s	✓ <1s
		7	✓ 42s	✓ 34m38s		✓ 1s	✓ <1s
		10	🛑	⏰		✓ 1s	✓ <1s
		12				✓ 4s	✓ 2s
		14				✓ 7s	✓ 2s
		17				✓ 12s	✓ 8s
	Needham-Schroeder-Lowe	2	✓ <1s	✓ <1s	✓ 30s		✓ <1s
		4	✓ 3s	BUG	⏰	✗	✓ <1s
		8	🛑				✓ 2s
		12					✓ 21s
	Otway-Rees	3	✓ 28s	✓ 2s	✓ 58m9s		✓ <1s
		6	🛑	🛑	⏰		✓ <1s
		7				✗	✓ <1s
		10					✓ 3s
		14					✓ 5m28s

✓ successful verification ϟ attack found ✗ out of scope
🛑 out of memory/stack overflow ⏰ timeout (12 hours)

Figure 15: Benchmark results (Part 1)

do not directly compute and return the children of each node in the most straightforward manner, but proceed in two steps:

1. We start with a breadth-first generation of the partition tree. The number of pending nodes will gradually grow until, potentially, exceeding a threshold parameter n.

2. Each available core focuses on one of these nodes, computes the whole subtree rooted by this node (depth-first manner), and is then assigned a new node. If at some point cores become idle—because all nodes generated at step 1 are either completed or currently assigned to an active core—we restart this two-step procedure on incomplete nodes.

Protocol (# of roles)			Akiss	APTE	SPEC	Sat-Eq	DeepSec
Anonymity	Private Authentication	2	✓ <1s	✓ <1s	✗	✗	✓ <1s
		4	✓ <1s	✓ 1s			✓ <1s
		6	✓ 21s	✓ 4m18s			✓ <1s
		8	OM	⏱			✓ 1s
		10					✓ 2s
		15					✓ 32s
	3G-AKA	4	✓ 1m34s	✓ 1h38m	✗	✗	✓ <1s
		6	OM	⏱			✓ 3s
	Passive Authentication	2	✓ <1s	✓ <1s	✓ 3s	✗	✓ <1s
		4	✓ <1s	✓ 1s	✓ 25m27s		✓ <1s
		6	✓ 2m22s	✓ 1m26s	OM		✓ <1s
		7	✓ 1h42m	✓ 1m40s			✓ 1s
		9	⏱	✓ 1h55m			✓ <1s
		15		⏱			✓ 4s
		21					✓ 8s
Unlinkability	3G-AKA	4	✓ 1m35s	✓ 1h23m	✗	✗	✓ <1s
		6	OM	⏱			✓ 2s
	Passive Authentication	4	✓ <1s	✓ 1s	✓ 25m31s	✗	✓ <1s
		6	✓ 2m15s	✓ 1m27s	OM		✓ <1s
		7	✓ 1h40m	✓ 1m44s			✓ 1s
		9	⏱	✓ 2h08m			✓ <1s
		15		⏱			✓ 9s
		21					✓ 15s
	BAC	4	OM	⚡ 38m56s	✗	✗	⚡ 1s
		6		⏱			⏱
Vote privacy	Prêt-à-Voter	6	✗	✗	✗	✗	✓ 2s
	Helios Vanilla	6	⚡ 47s	⚡ <1s	✗	✗	⚡ <1s
	Helios NR-W	6	OM	✗	✗	✗	✓ 1s
	Helios NR-ZKP	6					✓ 2s
	Helios dR-W	10					✓ 30m24s
	Helios dR-ZKP	10					✓ 9m26s
	Helios hR-W	11					⚡ 2s
	Helios hR-ZKP	11					✓ 2h42m

✓ successful verification ⚡ attack found ✗ out of scope
OM out of memory/stack overflow ⏱ timeout (12 hours)

Figure 16: Benchmark results (Part 2)

While parallelisation is also supported by the Akiss tool, DeepSec goes one step further as it is also able to distribute the computation through clusters of computers.

Benchmarks We performed extensive benchmarks to compare our tool against other tools that verify equivalence properties for a bounded number of sessions: Akiss [CCCK16], Apte [Che14], SAT-Equiv [CDD17] and Spec [TNH16]. Experiments are carried out on Intel Xeon 3.10GHz cores, with 40Go of memory. Akiss and DeepSec use 35 cores as they support parallelisation—unlike the others which therefore use a single core. The results are summarised in figs. 15 and 16.

We analysed strong secrecy, an equivalence based version of secrecy, for several classical authentication protocols. These benchmarks are mainly used for measuring scalability when increasing the number of sessions (figs. 15 and 16 indicate the number of roles in parallel, as depending on the exact scenario a session may require more or less roles). The Deepsec tool clearly outperforms Akiss, Apte, and Spec. The SAT-Equiv tool becomes more efficient, when the number of sessions significantly increases. However, the Otway-Rees protocol cannot be analysed by SAT-Equiv as it does not satisfy their type compliance condition and the Needham-Schroeder-Lowe protocol is out of its scope, as SAT-Equiv does not support asymmetric encryption.

To illustrate the broad scope of the tool we analyse unlinkability and anonymity properties for a number of other protocols: Abadi and Fournet's anonymous authentication protocol [AF04], the AKA protocol deployed in 3G telephony networks [AMR^{+}12], the Passive Authentication and Basic Access Control (BAC) protocols implemented in the European passport [For04], as well as the Prêt-à-Voter (PaV) [RS06] and several variants of the mixnet based Helios [Adi08] voting protocols. We comment a bit more on the voting protocol examples. Relying on the reduction result of Arapinis et al. [ACK16], we know that it is sufficient to consider three voters, two honest and one dishonest one, to conclude vote privacy for an arbitrary number of voters. Moreover, when revoting is allowed, which is the case for Helios, but not for PaV, we only need to consider a server which accepts seven ballots that may come from any of the three honest voters. For the Helios protocol we consider several versions. The *vanilla* Helios version, which does not allow revoting, is known to be vulnerable to a ballot-copy attack [CS13]—the attacker simply copies the ballot of a honest voter in order to bias the outcome. Two countermeasures have been proposed to thwart this attack: one applies a ballot weeding procedure (W), while the other is based on a zero-knowledge proof (ZKP) that links the identity of the voter to the ballot. When no revote (NR) is allowed these two versions are indeed shown to be secure. When allowing revoting we consider the case where seven ballots can be accepted [ACK16], under two different scenarios. When only the dishonest voter revotes (dR) we can show the security of the weeding mechanism. When however one honest voter re-votes twice (the same vote), a variant of the ballot-copy, pointed out to us by Rønne [Røn16], is possible in the weeding version. The attacker intercepts or delays the first honest vote, and casts this ballot in his name. The same ballot by the honest voter is than removed through weeding. However, as the honest voter casts a second (differently randomised) ballot the tally is biased by containing an additional vote for the honest voter's candidate. This attack is found by Deepsec. We can show that the ZKP version does not suffer from this attack as ballots are linked to voter identities and cannot be cast on behalf of someone else. Besides note that, while PaV is a priori in the scope of Akiss, it failed to produce a proof: Akiss only approximates trace equivalence of non-determinate processes and finds a false attack here.

Finally we note that BAC, Prêt-à-Voter and Helios protocols are not action-determinate and therefore do not benefit from the POR optimisation, which explains the much higher verification times when increasing the sessions. Nevertheless, as exemplified by the Helios hR-W example, attacks may be found very efficiently, as it generally does not require to explore the entire state space.

16 Conclusion and future work

In this paper we have studied automated verification of equivalence properties, encompassing both theoretical and practical aspects. We provide tight complexity results for static equivalence, trace equivalence and labelled bisimilarity, summarised in fig. 1. To obtain complexity upper bounds we propose a new decision procedure. In particular we show that deciding trace equivalence and labelled bisimilarity for a bounded number of sessions is coNEXP complete for subterm convergent destructor rewrite systems. Finally, we implement our procedure in the DEEPSEC prototype. As demonstrated through an extensive benchmark (figs. 15 and 16), our tool is broad in scope and efficient compared to other tools.

Our work opens several directions for future work. First, we wish to lift the restriction of subterm convergent equational theories. Even though the problem becomes quickly undecidable for more general rewrite theories, we plan to design a partially correct, i.e., sound, complete, but not necessarily terminating, procedure, as the procedure underlying the AKISS tool [CCCK16]. Second, we plan to avoid the restriction to destructor rewrite systems to more general ones. From the complexity point of view we envision to study *parametrised* complexity (taking the rewrite system, or the degree of non-determinism as a parameter). This may increase our understanding of which parts of the input are responsible of the high complexity, and guide further optimisations. We have seen that the POR techniques have dramatically increased the tool's performances on action-determinate processes. We wish to develop similar techniques for more general classes of processes.

References

[ABB+05] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*, Lecture Notes in Computer Science, pages 281–285. Springer, 2005.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, November 2006.

[ACK16] Myrto Arapinis, Véronique Cortier, and Steve Kremer. When are three voters enough for privacy properties? In *Proceedings of the 21st European Symposium on Research in Computer Security (ESORICS'16)*, volume 9879 of *Lecture Notes in Computer Science*, pages 241–260. Springer, September 2016.

[ACRR10] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark D. Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Comp. Soc. Press, 2010.

[Adi08] B. Adida. Helios: web-based open-audit voting. In *17th conference on Security symposium (SS'08)*, pages 335–348. USENIX Association, 2008.

[AF01] Martín Abadi and Cédric Fournet. Mobile Values, New Names, and Secure Communication. In *28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, September 2004.

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999.

[AMR$^+$12] M. Arapinis, L. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *19th Conference on Computer and Communications Security (CCS'12)*, pages 205–216. ACM Press, 2012.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science (LICS'05)*, pages 331–340. IEEE Comp. Soc. Press, June 2005.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *12th Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM Press, 2005.

[BDH15] David Baelde, Stéphanie Delaune, and Lucca Hirschi. Partial order reduction for security protocols. In *26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 497–510. Leibniz-Zentrum für Informatik, September 2015.

[BDS15] David A. Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In *22nd Conference on Computer and Communications Security (CCS'15)*, pages 1144–1155. ACM Press, 2015.

[BS16] Bruno Blanchet and Ben Smyth. Automated reasoning for equivalences in the applied pi calculus with barriers. In *29th Computer Security Foundations Symposium (CSF'16)*, pages 310–324. IEEE Comp. Soc. Press, 2016.

[BSC16] Bruno Blanchet, Ben Smyth, and Vincent Cheval. *Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2016.

[BT00] Michele Boreale and Luca Trevisan. A complexity analysis of bisimilarity for value-passing processes. *Theor. Comput. Sci.*, 238(1-2):313–345, 2000.

[CB13] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with ProVerif. In *Proc. 2nd Conference on Principles of Security and Trust (POST'13)*, volume 7796 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2013.

[CBC10] Bruno Conchinha, David A. Basin, and Carlos Caleiro. Efficient decision procedures for message deducibility and static equivalence. In *Proc. 7th International Workshop on Formal Aspects of Security and Trust (FAST'10)*, volume 6561 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2010.

[CCCK16] Rohit Chadha, Vincent Cheval, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocol. *ACM Transactions on Computational Logic*, 23(4):1–32, 2016.

[CCD13] Vincent Cheval, Véronique Cortier, and Stéphanie Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 492:1–39, April 2013.

[CCD15a] Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Decidability of trace equivalence for protocols with nonces. In *Proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF'15)*, pages 170–184. IEEE Computer Society Press, 2015.

[CCD15b] Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. From security protocols to pushdown automata. *ACM Transactions on Computational Logic*, 17(3):1–45, November 2015.

[CD05] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In *16th International Conference on Rewriting Techniques and Applications (RTA'05)*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

[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)*, pages 481–494. IEEE Computer Society Press, 2017.

[CDK12] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. *Journal of Automated Reasoning*, 48(2):219–262, 2012.

[CGLM17] Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. A type system for privacy properties. In *24th ACM Conference on Computer and Communications Security (CCS'17)*, Dallas, USA, October 2017. ACM. To appear.

[Che14] Vincent Cheval. Apte: an algorithm for proving trace equivalence. In *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *Lecture Notes in Computer Science*, pages 587–592. Springer, 2014.

[CR10] Yannick Chevalier and Michaël Rusinowitch. Decidability of equivalence of symbolic derivations. *Journal of Automated Reasoning*, 48(2):263–292, 2010.

[Cre08] Cas J. F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Proc. 20th International Conference on Computer Aided Verification (CAV'08)*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

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

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[DLM04] Nancy A. Durgin, Patrick Lincoln, and John C. Mitchell. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12(2):247–311, 2004.

[DY81] D. Dolev and A.C. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of Computer Science (FOCS'81)*, pages 350–357. IEEE Comp. Soc. Press, 1981.

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

[HBD16] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for verifying privacy-type properties: the unbounded case. In *37th IEEE Symposium on Security and Privacy (S&P'16)*, pages 564–581, San Jose, California, USA, 2016. IEEE Comp. Soc. Press.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, 1992.

[Pap03] C.H. Papadimitriou. *Computational complexity.* John Wiley and Sons Ltd., 2003.

[Røn16] P. Rønne. Personal communication, 2016.

[RS06] Peter Y. A. Ryan and Steve A. Schneider. Prêt-à-voter with re-encryption mixes. In *11th European Symp. On Research In Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326. Springer, 2006.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions, composed keys is np-complete. *Theor. Comput. Sci.*, 299(1-3):451–475, 2003.

[SEMM14] Sonia Santiago, Santiago Escobar, Catherine Meadows, and José Meseguer. A formal definition of protocol indistinguishability and its verification using Maude-NPA. In *10th International Workshop on Security and Trust Management STM'14*, volume 8743 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2014.

[SMCB13] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *25th International Conference on Computer Aided Verification (CAV'13)*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

[TD10] Alwen Tiu and Jeremy Dawson. Automating open bisimulation checking for the spi-calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 307–321. IEEE Comp. Soc. Press, 2010.

[TNH16] Alwen Tiu, Nam Nguyen, and Ross Horne. SPEC: an equivalence checker for security protocols. In *14th Asian Symposium on Programming Languages and Systems (APLAS'16)*, volume 10017 of *Lecture Notes in Computer Science*, pages 87–95. Springer, 2016.

Part IV

Appendix

A Proofs of Part I (lower bounds)

A.1 Advanced winning strategies

Before starting the proofs, we present some characterizations of observational (in)equivalence in order to make the incoming proofs easier to handle. It somehow formalizes the intuition that observational equivalence is a bisimulation game (see section 4.2).

Remark 10. The results of this section (A.1) also apply to the extended semantics of section 3.

A.1.1 For the defender

The transitions of the semantics which are deterministic and silent are not essential to equivalence proofs as they do not interfere substantially with them. We introduce below a refined proof technique to rule them out.

Definition 31 (simplification)**.** *A multiset of closed plain processes $\mathcal{S}$ is silent in an extended process $(\mathcal{P}, \Phi)$ when for all transitions $(\mathcal{P} \cup \mathcal{S}, \Phi) \xrightarrow{\alpha}_{\mathsf{c}} (\mathcal{Q}, \Phi')$, it holds that $\mathcal{Q} = \mathcal{P}' \cup \mathcal{S}$ with $(\mathcal{P}, \Phi) \xrightarrow{\alpha}_{\mathsf{c}} (\mathcal{P}', \Phi')$ and $\mathcal{S}$ silent in $(\mathcal{P}', \Phi')$. Then we define $\rightsquigarrow$ (simplification relation) the relation on extended processes defined by the following inference rules:*

$$\frac{\mathcal{S} \textit{ silent in } (\mathcal{P}, \Phi)}{(\mathcal{P} \cup \mathcal{S}, \Phi) \rightsquigarrow (\mathcal{P}, \Phi)} \quad (\text{S-}\textsc{sil})$$

$$\frac{c \in \mathcal{N}_{\mathsf{prv}} \qquad Msg(t) \qquad c \notin names(\mathcal{P}, \Phi)}{(\mathcal{P} \cup \{\!\{\overline{c}\langle t\rangle.P, c(x).Q\}\!\}, \Phi) \rightsquigarrow (\mathcal{P} \cup \{\!\{P, Q\{x \mapsto t\}\}\!\}, \Phi)} \quad (\text{S-}\textsc{comm})$$

$$\frac{A \xrightarrow{\varepsilon}_{\mathsf{c}} B \textit{ by rules } \textsc{Null}, \textsc{Par}, \textsc{Then}, \textsc{Else}}{A \rightsquigarrow B} \quad (\text{S-}\textsc{npte})$$

In other words, we write $A \rightsquigarrow B$ when B is obtained from A by removing some silent process or applying a deterministic (in the sense of the confluence lemma below) instance of the transition relation $\xrightarrow{\varepsilon}_{\mathsf{c}}$. We call $\rightsquigarrow_{\mathsf{pi}}$ the restriction of $\rightsquigarrow$ to the rule (S-npte). Their reflexive transitive closures are denoted $\overset{\star}{\rightsquigarrow}$ and $\overset{\star}{\rightsquigarrow}_{\mathsf{pi}}$ respectively as usual.

Lemma 17. *If $A \rightsquigarrow B$ (by some rule ρ_{sil} of the definition of $\rightsquigarrow$) and $A \xrightarrow{\alpha}_{\mathsf{c}} C$ (by some rule ρ_{c} of the semantics), then either $B = C$ and $\alpha = \varepsilon$, or there exists D such that $C \rightsquigarrow D$ (by rule ρ_{sil}) and $B \xrightarrow{\alpha}_{\mathsf{c}} D$ (by rule ρ_{c}).*

Proof. We make a case analysis on the rule used to obtain the reduction $A \rightsquigarrow B$:

- **case 1** : by rule (S-sil):

  Then we write $A = (\mathcal{P} \cup \mathcal{S}, \Phi)$, $B = (\mathcal{P}, \Phi)$. By definition of silent processes, the reduction $A \xrightarrow{\alpha}_{\mathsf{c}} C$ hence gives $C = (\mathcal{P}' \cup \mathcal{S}, \Phi')$ where $(\mathcal{P}, \Phi) \xrightarrow{\alpha}_{\mathsf{c}} (\mathcal{P}', \Phi') = D$ and $\mathcal{S}$ silent in D. In particular D gives the expected conclusion.

· **case 2** : by rule (S-COMM) or (S-NPTE):

Then either $B = C$ and the conclusion is immediate, or $B \neq C$ and a quick analysis of the rules of the semantics gives $\mathcal{P}, Q_B, Q'_B, Q_C, Q'_C, \Phi, \Phi'$ such that:

$$A = (\{\!\{Q_B, Q_C\}\!\} \cup \mathcal{P}, \Phi) \quad B = (\{\!\{Q'_B, Q_C\}\!\} \cup \mathcal{P}, \Phi) \qquad C = (\{\!\{Q_B, Q'_C\}\!\} \cup \mathcal{P}, \Phi')$$
$$(\{\!\{Q_B\}\!\}, \Phi) \rightsquigarrow (\{\!\{Q'_B\}\!\}, \Phi) \qquad (\{\!\{Q_C\}\!\}, \Phi) \xrightarrow{\alpha}_{\mathsf{c}} (\{\!\{Q'_C\}\!\}, \Phi')$$

and we conclude by choosing $D = (\{\!\{Q'_B, Q'_C\}\!\} \cup \mathcal{P}, \Phi')$. □

Corollary 2. *If $A \overset{\star}{\rightsquigarrow} B$ and $A \xrightarrow{\alpha}_{\mathsf{c}} C$ then either $B \overset{\star}{\rightsquigarrow} C$ and $\alpha = \varepsilon$, or there exists D such that $C \overset{\star}{\rightsquigarrow} D$ and $B \xrightarrow{\alpha}_{\mathsf{c}} D$.*

Proof. By a straightforward induction on the number of steps of the reduction $A \overset{\star}{\rightsquigarrow} B$. □

Corollary 3. *$\rightsquigarrow_{\mathsf{pi}}$ is convergent.*

Proof. The termination of $\rightsquigarrow_{\mathsf{pi}}$ follows from the termination of the whole calculus. As for the local confluence (which sufficies by Newmann's lemma), we observe that by lemma 17, if $A \rightsquigarrow_{\mathsf{pi}} B$ and $A \rightsquigarrow_{\mathsf{pi}} C$ then either $B = C$, or there is D such that $B \rightsquigarrow_{\mathsf{pi}} D$ and $C \rightsquigarrow_{\mathsf{pi}} D$: in particular, $B \overset{\star}{\rightsquigarrow}_{\mathsf{pi}} E$ and $C \overset{\star}{\rightsquigarrow}_{\mathsf{pi}} E$ for some $E \in \{C, D\}$. □

In particular, all extended processes A have a unique normal form w.r.t. $\rightsquigarrow_{\mathsf{pi}}$ which will be written $A{\downarrow_{\mathsf{pi}}}$. This notation is lifted to multiset of processes, writing $\mathcal{P}{\downarrow_{\mathsf{pi}}}$ (which is consistent since $\rightsquigarrow_{\mathsf{pi}}$ does not modify the frame). With all of this, we eventually gathered all the ingredients to introduce our characterization of bisimilarity:

Definition 32 (bisimulation up to $\rightsquigarrow$)**.** *A symmetric relation $\mathscr{R}$ on extended processes is then said to be bisimulation up to $\rightsquigarrow$, or a bisimulation up to simplification, when:*

- *$\mathscr{R} \subseteq \sim$;*
- *for all extended processes A, B such that $A \mathscr{R} B$, and for all transitions $A \xrightarrow{\alpha}_{\mathsf{c}} A'$, there exists $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$ such that $A' \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} B'$.*

Proposition 1. *For all extended processes A and B, $A \approx_o B$* **iff** *there exists a bisimulation up to simplification $\mathscr{R}$ such that $A \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} B$.*

Proof. The forward implication follows from the fact that $\approx_o$ is a bisimulation up to simplification (by reflexivity of $\overset{\star}{\rightsquigarrow}$). For the converse, let us consider $\mathscr{R}$ a bisimulation up to $\rightsquigarrow$ and prove that it is contained in $\approx_o$. In order to do that, it sufficies to show that:

- $\overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow}$ is symmetric;
- $(\overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow}) \subseteq \sim$;
- for all extended processes A, B such that $A \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} B$, if $A \xrightarrow{\alpha}_{\mathsf{c}} A'$ then there exists $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$ such that $A' \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} B'$.

These three properties indeed justify that $(\overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow}) \subseteq \approx_o$ by definition, hence the expected conclusion as $\mathscr{R} \subseteq \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow}$ by reflexivity of $\overset{\star}{\rightsquigarrow}$. Yet it appears that the first two points directly follows from the properties of $\mathscr{R}$ and the reflexivity of $\overset{\star}{\rightsquigarrow}$, and we thus only need to prove the third point. Let us therefore consider the following hypotheses and notations:

$$A \overset{\star}{\rightsquigarrow} C \mathscr{R} D \overset{\star}{\leftsquigarrow} B \qquad\qquad A \xrightarrow{\alpha}_{\mathsf{c}} A'$$

and let us exhibit B' such that $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$ and $A' \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} B'$. Let us consider the two cases induced by the application of corollary 2:

· **case 1** : $A' \overset{\star}{\rightsquigarrow} C$ and $\alpha = \varepsilon$

Then we can choose $B' = B$.

· **case 2** : there exists C' such that $A' \overset{\star}{\rightsquigarrow} C'$ and $C \xrightarrow{\alpha}_{\mathsf{c}} C'$

Consequently, since $\mathscr{R}$ is a bisimulation up to simplification, there is D' such that $D \overset{\alpha}{\Rightarrow}_{\mathsf{c}} D'$ and $C' \overset{\star}{\rightsquigarrow} \mathscr{R} \overset{\star}{\leftsquigarrow} D'$. Then we remark that for all extended processes B_1, B_2, B_3:

- a transition $B_1 \rightsquigarrow B_2$ with rules (S-NPTE) or (S-COMM) implies $B_1 \xrightarrow{\varepsilon}_{\mathsf{c}} B_2$;
- if $B_1 \rightsquigarrow B_2$ with rule (S-SIL) and $B_2 \xrightarrow{\beta}_{\mathsf{c}} B_3$, then $B_1 \xrightarrow{\beta}_{\mathsf{c}} \rightsquigarrow B_3$.

In particular since $B \overset{\star}{\rightsquigarrow} D \overset{\alpha}{\Rightarrow}_{\mathsf{c}} D'$, we have $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} D'' \overset{\star}{\rightsquigarrow} D'$ for some D''. Hence the conclusion by choosing $B' = D''$. □

A.1.2 For the attacker

When taking the negation of labelled bisimilarity, we essentially obtain a set of rules for a game whose states are pairs of processes (A, B): an attacker selects a transition and a defender answers by selecting a equivalently-labelled sequence of transitions in the other process.

Definition 33 (labelled attack). *A relation $\mathscr{S}$ on extended processes is called a labelled attack when for all A, B such that $A \mathscr{S} B$, it holds that:*

1. *either: $A \not\sim B$*
2. *or: $\exists A \xrightarrow{\alpha}_{\mathsf{c}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A'$, $\forall B \overset{\alpha.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} B'$, $A' \mathscr{S} B'$*
3. *or: $\exists B \xrightarrow{\alpha}_{\mathsf{c}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} B'$, $\forall A \overset{\alpha.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} A'$, $A' \mathscr{S} B'$*

Note that labelled attacks are not the direct translation of the above intuition since they allow the attacker to choose several transitions in a row; this intuitively entails no loss of generality since it is equivalent to the attacker selecting some transitions non-adaptatively (i.e. independently of the answer of the defender). Here is the formal statement of correctness:

Proposition 2. *For all extended processes A and B, $A \not\approx_o B$* **iff** *there exists a labelled attack $\mathscr{S}$ such that $A \mathscr{S} B$.*

Proof. The forward implication is immediate since $\not\approx_o$ is a labelled attack (we can even choose $\mathsf{tr} = \varepsilon$ everytime). Let then $\mathscr{S}$ be a labelled attack such that $A \mathscr{S} B$ and let us prove that $\mathscr{S} \subseteq \not\approx_o$. More precisely, we prove that $\mathscr{S} \subseteq \mathscr{S}' \subseteq \not\approx_o$ for some relation $\mathscr{S}'$. We will construct $\mathscr{S}'$ in such a way that for all A, B extended processes, $A \mathscr{S}' B$ entails:

(i) either: $A \not\sim B$;

(ii) or: $\exists A \xrightarrow{\alpha}_{\mathsf{c}} A',\ \forall B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B',\ A' \mathscr{S} B'$;

(iii) or: $\exists B \xrightarrow{\alpha}_{\mathsf{c}} B',\ \forall A \overset{\alpha}{\Rightarrow}_{\mathsf{c}} A',\ A' \mathscr{S} B'$

The inclusion $\mathscr{S}' \subseteq \not\approx_o$ is indeed clear if this property is verified, hence the expected conclusion provided such a relation $\mathscr{S}'$. We concretely define it as the smallest relation on extended processes saturated by the following inference rules:

$$\frac{A \mathscr{S} B}{A \mathscr{S}' B}\ (\textsc{Axiom})$$

$$\frac{A \mathscr{S}' B \qquad A \xrightarrow{\alpha}_{\mathsf{c}} A' \xrightarrow{\alpha'}_{\mathsf{c}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A'' \qquad \forall B \overset{\alpha.\alpha'.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} B'', A'' \mathscr{S}' B'' \qquad B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'}{A' \mathscr{S}' B'}\ (\textsc{Dec-L})$$

$$\frac{A \mathscr{S}' B \qquad B \xrightarrow{\alpha}_{\mathsf{c}} B' \xrightarrow{\alpha'}_{\mathsf{c}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} B'' \qquad \forall A \overset{\alpha.\alpha'.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} A'', A'' \mathscr{S}' B'' \qquad A \overset{\alpha}{\Rightarrow}_{\mathsf{c}} A'}{A' \mathscr{S}' B'}\ (\textsc{Dec-R})$$

In particular, note that $\mathscr{S} \subseteq \mathscr{S}'$ thanks to the rule (Axiom). As for the two other rules (Dec-L) and (Dec-R), they intuitively decompose sequences $\xrightarrow{\alpha}_{\mathsf{c}} \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}}$ into atomic transitions in order to switch from points *2.* or *3.* of definition 33 to points (ii) or (iii).

Let then A and B be two extended processes such that $A \mathscr{S}' B$. We consider a proof-tree of $A \mathscr{S}' B$ in the inference system above and perform a case analysis on the rule at its root:

- **case 1** (Axiom): $A \mathscr{S} B$.

  As $\mathscr{S}$ is a labelled attack, we apply the case analysis of definition 33:

  - **case 1.a** : $A \not\sim B$.
    Then (i) is satisfied.
  - **case 1.b** : there exists $A \xrightarrow{\alpha}_{\mathsf{c}} A' \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A''$ such that $A'' \mathscr{S} B''$ for all $B \overset{\alpha.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} B''$.
    In particular, keeping in mind that $\mathscr{S} \subseteq \mathscr{S}'$ due to the rule (Axiom), we have $A'' \mathscr{S}' B''$ for all $B \overset{\alpha.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} B''$. Let us then show that the transition $A \xrightarrow{\alpha}_{\mathsf{c}} A'$ satisfies (ii). We therefore have to show that $A' \mathscr{S}' B'$ for all $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$. If $A' = A''$ then the result follows from the hypothesis. Otherwise let us write $A \xrightarrow{\alpha}_{\mathsf{c}} A' \xrightarrow{\alpha'}_{\mathsf{c}} \overset{\mathsf{tr}'}{\Rightarrow}_{\mathsf{c}} A''$ where $\alpha'.\mathsf{tr}' = \mathsf{tr}$ and the rule (Dec-L) justifies that $A' \mathscr{S}' B'$ for all $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$.
  - **case 1.c** : there exists $B \xrightarrow{\alpha}_{\mathsf{c}} B' \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} B''$ such that $B'' \mathscr{S} A''$ for all $A \overset{\alpha.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} A''$.
    Analogous, targeting (iii) instead of (ii) and replacing (Dec-L) by (Dec-R).

- **case 2** (Dec-L): there are A_0, A_1, A_2, B_0, B_2, α, α', tr, such that $A_0 \mathscr{S}' B_0$, $B_0 \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B$, $A_0 \xrightarrow{\alpha}_{\mathsf{c}} A \xrightarrow{\alpha'}_{\mathsf{c}} A_1 \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} A_2$, and $\forall\ B_0 \overset{\alpha.\alpha'.\mathsf{tr}}{\Longrightarrow}_{\mathsf{c}} B_2, A_2 \mathscr{S}' B_2$.

  Let us show that the transition $A \xrightarrow{\alpha'}_{\mathsf{c}} A_1$ satisfies (ii). We therefore have to show that $A_1 \mathscr{S}' B_1$ for all $B \overset{\alpha'}{\Rightarrow}_{\mathsf{c}} B_1$. If $A_1 = A_2$ then the result follows from the hypothesis.

Otherwise we write $A \xrightarrow{\alpha'}_{\mathsf{c}} A_1 \xrightarrow{\alpha''}_{\mathsf{c}} \overset{\mathsf{tr}'}{\Rightarrow}_{\mathsf{c}} A_2$ where $\alpha''.\mathsf{tr}' = \mathsf{tr}$ and the rule (Dec-L) justifies that $A_1 \mathscr{S}' B_1$ for all $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B_1$.

· **case 3** (Dec-R): Analogous to case 2. □

A.2 Correctness of the encodings (section 3)

Now we prove that the translation $[\![\cdot]\!]$ of the extended semantics is correct (lemma 1). But first of all, a (trivial) observation about the free variables of a translated process:

Lemma 18. *For all plain processes P and all first-order substitution σ, $[\![P\sigma]\!] = [\![P]\!]\sigma$.*

We will use this lemma implicitly in the remaining of this section. Besides, as we have the inclusion of relations $\approx_o \subseteq \approx_t$, we only need to prove the observational-equivalence statement of lemma 1. We recall that we use notations $A{\downarrow_{\mathsf{pi}}}$ and $\mathcal{P}{\downarrow_{\mathsf{pi}}}$ to refer to normal forms w.r.t. $\rightsquigarrow_{\mathsf{pi}}$ (see corollary 3).

Proposition 3. *We consider $\mathscr{R}$ the symmetric closure of:*

$$\{(C, [\![C]\!]{\downarrow_{\mathsf{pi}}}) \mid C \textit{ extended process such that } C = C{\downarrow_{\mathsf{pi}}}\}$$

$\mathscr{R}$ is a bisimulation up to simplification.

Proof. $\mathscr{R}$ is symmetric by definition and is trivially included in $\sim$. Let then $(A, B) \in \mathscr{R}$ and $A \xrightarrow{\alpha}_{\mathsf{c}} A'$ and let us exhibit B' such that $B \overset{\alpha}{\Rightarrow}_{\mathsf{c}} B'$ and $A'\mathscr{R}B'$. We perform a case analysis on the rule triggerring the transition $A \xrightarrow{\alpha}_{\mathsf{c}} A'$:

· **case 1** (rules Null, Par, Then, Else):

This case cannot arise as A is in normal form w.r.t. $\rightsquigarrow_{\mathsf{pi}}$ by definition of $\mathscr{R}$.

· **case 2** (rule In): $\alpha = \xi(\zeta)$ for some $\xi, \zeta \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup \mathit{dom}(\Phi))$ and:

$$\begin{array}{ll} A = (\mathcal{P} \cup \{\!\{u(x).P\}\!\}, \Phi) & \text{with } \mathit{Msg}(u),\ \mathit{Msg}(\xi\Phi) \text{ and } \xi\Phi{\downarrow} = u{\downarrow} \\ A' = (\mathcal{P} \cup \{\!\{P\{x \mapsto \zeta\Phi{\downarrow}\}\}\!\}, \Phi) & \text{with } \mathit{Msg}(\zeta\Phi) \end{array}$$

Then, by a case analysis on the hypothesis $A\mathscr{R}B$:

· **case 2.a** : $B = [\![A]\!]{\downarrow_{\mathsf{pi}}}$
Then we can write:

$$B = ([\![\mathcal{P}]\!]{\downarrow_{\mathsf{pi}}} \cup \{\!\{u(x).[\![P]\!]\}\!\}, \Phi)$$

and we conclude by remarking that $B \xrightarrow{\xi(\zeta)}_{\mathsf{c}} B' = ([\![\mathcal{P}]\!]{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![P]\!]\{x \mapsto \zeta\Phi{\downarrow}\}\}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} A'{\downarrow_{\mathsf{pi}}} \mathscr{R} [\![A'{\downarrow_{\mathsf{pi}}}]\!]{\downarrow_{\mathsf{pi}}} = [\![A']\!]{\downarrow_{\mathsf{pi}}} = ([\![\mathcal{P}]\!]{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![P]\!]\{x \mapsto \zeta\Phi{\downarrow}\}\}\!\}{\downarrow_{\mathsf{pi}}}, \Phi) \overset{\star}{\leftsquigarrow} B'$$

· **case 2.b** : $A = [\![B]\!]{\downarrow_{\mathsf{pi}}}$ (and $B = B{\downarrow_{\mathsf{pi}}}$)

Note that $u = \xi\Phi\!\downarrow$ cannot be one of the names introduced by the translation $[\![\cdot]\!]$: these names can indeed not appear in Φ since they are chosen private and fresh and since the semantics cannot introduce new private names. In particular we can write:

$$B = (\mathcal{Q} \cup \{\!\{u(x).Q\}\!\}, \Phi) \qquad \text{with } [\![Q]\!] = P \text{ and } [\![\mathcal{Q}]\!]_{\downarrow_{\mathsf{pi}}} = \mathcal{P}$$

and we conclude by writing $B \xrightarrow{\xi(\zeta)}_{\mathsf{c}} B' = (\mathcal{Q} \cup \{\!\{Q\{x \mapsto \zeta\Phi\!\downarrow\}\}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} ([\![\mathcal{Q}]\!]_{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![Q]\!]\{x \mapsto \zeta\Phi\!\downarrow\}\}\!\}_{\downarrow_{\mathsf{pi}}}, \Phi) = [\![B']\!]_{\downarrow_{\mathsf{pi}}} = [\![B'_{\downarrow_{\mathsf{pi}}}]\!]_{\downarrow_{\mathsf{pi}}} \mathscr{R}\, B'_{\downarrow_{\mathsf{pi}}} \overset{\star}{\leftsquigarrow} B'$$

- **case 3** (rule Out): $\alpha = \overline{\xi}\langle \mathsf{ax}_n \rangle$ for some $\xi, \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{\mathsf{pub}} \cup dom(\Phi))$, $\mathsf{ax}_n \in \mathcal{AX}$ and:

$$\begin{aligned} A &= (\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P\}\!\}, \Phi) && \text{with } Msg(u),\ Msg(t),\ Msg(\xi\Phi) \text{ and } \xi\Phi\!\downarrow = u\!\downarrow \\ A' &= (\mathcal{P} \cup \{\!\{P\}\!\}, \Phi') && \text{where } \Phi' = \Phi \cup \{\mathsf{ax}_n \mapsto t\!\downarrow\} \text{ and } n = |\Phi| + 1 \end{aligned}$$

Then, by a case analysis on the hypothesis $A\mathscr{R}B$:

  - **case 3.a** : $B = [\![A]\!]_{\downarrow_{\mathsf{pi}}}$
  Then we can write:

$$B = ([\![\mathcal{P}]\!]_{\downarrow_{\mathsf{pi}}} \cup \{\!\{\overline{u}\langle t\rangle.[\![P]\!]\}\!\}, \Phi)$$

  and we conclude by remarking that $B \xrightarrow{\overline{\xi}\langle \mathsf{ax}_n\rangle}_{\mathsf{c}} B' = ([\![\mathcal{P}]\!]_{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![P]\!]\}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} A'_{\downarrow_{\mathsf{pi}}} \mathscr{R}\, [\![A'_{\downarrow_{\mathsf{pi}}}]\!]_{\downarrow_{\mathsf{pi}}} = [\![A']\!]_{\downarrow_{\mathsf{pi}}} = ([\![\mathcal{P}]\!]_{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![P]\!]\}\!\}_{\downarrow_{\mathsf{pi}}}, \Phi) \overset{\star}{\leftsquigarrow} B'$$

  - **case 3.b** : $A = [\![B]\!]_{\downarrow_{\mathsf{pi}}}$ (and $B = B_{\downarrow_{\mathsf{pi}}}$)
  For the same reason as in case 2.b, we can write:

$$B = (\mathcal{Q} \cup \{\!\{\overline{u}\langle t\rangle.Q\}\!\}, \Phi) \qquad \text{with } [\![Q]\!] = P \text{ and } [\![\mathcal{Q}]\!]_{\downarrow_{\mathsf{pi}}} = \mathcal{P}$$

  and we conclude by writing $B \xrightarrow{\overline{\xi}\langle \mathsf{ax}_n\rangle}_{\mathsf{c}} B' = (\mathcal{Q} \cup \{\!\{Q\}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} ([\![\mathcal{Q}]\!]_{\downarrow_{\mathsf{pi}}} \cup \{\!\{[\![Q]\!]\}\!\}_{\downarrow_{\mathsf{pi}}}, \Phi) = [\![B']\!]_{\downarrow_{\mathsf{pi}}} = [\![B'_{\downarrow_{\mathsf{pi}}}]\!]_{\downarrow_{\mathsf{pi}}} \mathscr{R}\, B'_{\downarrow_{\mathsf{pi}}} \overset{\star}{\leftsquigarrow} B'$$

- **case 4** (rule (Comm)): $\alpha = \varepsilon$ and:

$$\begin{aligned} A &= (\mathcal{P} \cup \{\!\{\overline{u}\langle t\rangle.P, v(x).Q\}\!\}, \Phi) && \text{with } Msg(u),\ Msg(v),\ Msg(t) \text{ and } u\!\downarrow = v\!\downarrow \\ A' &= (\mathcal{P} \cup \{\!\{P, Q\{x \mapsto t\}\}\!\}, \Phi) \end{aligned}$$

Then, by a case analysis on the hypothesis $A\mathscr{R}B$:

· **case 4.a** : $B = \llbracket A \rrbracket_{\downarrow\mathsf{pi}}$
Then we can write:

$$B = (\llbracket \mathcal{P} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \overline{u}\langle t\rangle.\llbracket P \rrbracket, v(x).\llbracket Q \rrbracket \}\!\}, \Phi)$$

and we conclude by remarking that $B \xrightarrow{\varepsilon}_{\mathsf{c}} B' = (\llbracket \mathcal{P} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket P \rrbracket, \llbracket Q \rrbracket\{x \mapsto t\} \}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} A'_{\downarrow\mathsf{pi}} \mathrel{\mathscr{R}} \llbracket A'_{\downarrow\mathsf{pi}} \rrbracket_{\downarrow\mathsf{pi}} = \llbracket A' \rrbracket_{\downarrow\mathsf{pi}} = (\llbracket \mathcal{P} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket P \rrbracket_{\downarrow\mathsf{pi}}, \llbracket Q \rrbracket\{x \mapsto t\}_{\downarrow\mathsf{pi}} \}\!\}, \Phi) \overset{\star}{\leftsquigarrow} B'$$

· **case 4.b** : $A = \llbracket B \rrbracket_{\downarrow\mathsf{pi}}$ (and $B = B_{\downarrow\mathsf{pi}}$) and a term w such that $w{\downarrow} = u{\downarrow}$ appears in B (syntactically)
In particular ${\downarrow}u$ is not a fresh name introduced by the translation $\llbracket\cdot\rrbracket$ and we can therefore write:

$$B = (\mathcal{P}' \cup \{\!\{ \overline{u}\langle t\rangle.P', v(x).Q' \}\!\}, \Phi) \qquad \text{with } \llbracket \mathcal{P}' \rrbracket_{\downarrow\mathsf{pi}} = \mathcal{P},\ \llbracket P' \rrbracket = P \text{ and } \llbracket Q' \rrbracket = Q$$

and we conclude by writing $B \xrightarrow{\varepsilon}_{\mathsf{c}} B' = (\mathcal{P}' \cup \{\!\{ P', Q'\{x \mapsto t\} \}\!\}, \Phi)$ and:

$$A' \overset{\star}{\rightsquigarrow} (\llbracket \mathcal{P}' \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket P' \rrbracket_{\downarrow\mathsf{pi}}, \llbracket Q' \rrbracket\{x \mapsto t\}_{\downarrow\mathsf{pi}} \}\!\}, \Phi) = \llbracket B' \rrbracket_{\downarrow\mathsf{pi}} = \llbracket B'_{\downarrow\mathsf{pi}} \rrbracket_{\downarrow\mathsf{pi}} \mathrel{\mathscr{R}} B'_{\downarrow\mathsf{pi}} \overset{\star}{\leftsquigarrow} B'$$

· **case 4.c** : $A = \llbracket B \rrbracket_{\downarrow\mathsf{pi}}$ (and $B = B_{\downarrow\mathsf{pi}}$) and there exists no term w appearing in B such that $w{\downarrow} = u{\downarrow}$ (syntactically)
Then u is a fresh name introduced by $\llbracket\cdot\rrbracket$. We consider the two disjoint cases where it was introduced for the translation of a sum or a circuit:

- If $u \in \mathcal{N}_{\mathsf{prv}}$ is a fresh name generated in order to translate a sum of B, or rephrased more formally:

$$\begin{aligned} B &= (\mathcal{Q} \cup \{\!\{ P' + Q' \}\!\}, \Phi) & \\ A &= (\llbracket \mathcal{Q} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \overline{u}\langle u\rangle, u(x).\llbracket P' \rrbracket, u(x).\llbracket Q' \rrbracket \}\!\}, \Phi) & &\text{where } x \in \mathcal{X}^1 \text{ but } x \notin \mathit{vars}(P', Q') \\ A' &= (\llbracket \mathcal{Q} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket R \rrbracket, u(x).\llbracket S \rrbracket \}\!\}, \Phi) & &\text{where } R, S \in \{\!\{ P', Q' \}\!\},\ R \neq S \end{aligned}$$

  We let $B' = (\mathcal{Q} \cup \{\!\{ R \}\!\}, \Phi)$ and remark that $B \xrightarrow{\varepsilon}_{\mathsf{c}} B'$ by the rule Choice-L (if $R = P'$ and $S = Q'$) or Choice-R (if $R = Q'$ and $S = P'$). Besides, let us observe that the name u does not appear in $\llbracket \mathcal{Q} \rrbracket_{\downarrow\mathsf{pi}}$, $\llbracket R \rrbracket$ nor Φ by construction of $\llbracket\cdot\rrbracket$ and that $u(x).\llbracket S \rrbracket$ is therefore easily seen to be silent in $A'' = (\llbracket \mathcal{Q} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket R \rrbracket \}\!\}, \Phi)$. In particular it entails that $A' \rightsquigarrow A''$ by the rule (S-sil) which gives the conclusion:

$$A' \rightsquigarrow A'' \overset{\star}{\rightsquigarrow} (\llbracket \mathcal{Q} \rrbracket_{\downarrow\mathsf{pi}} \cup \{\!\{ \llbracket R \rrbracket \}\!\}_{\downarrow\mathsf{pi}}, \Phi) = \llbracket B' \rrbracket_{\downarrow\mathsf{pi}} = \llbracket B'_{\downarrow\mathsf{pi}} \rrbracket_{\downarrow\mathsf{pi}} \mathrel{\mathscr{R}} B'_{\downarrow\mathsf{pi}} \overset{\star}{\leftsquigarrow} B'$$

- $u \in \mathcal{N}_{\mathsf{prv}}$ is a fresh name generated in order to translate a $\texttt{Choose}(x)$: this case can be handle analogously to the previous one.

- If $u \in \mathcal{N}_{\mathsf{prv}}$ is a fresh name generated in order to translate a circuit of B, or formally:

$$B = (\mathcal{Q} \cup \{\!\{\vec{x} \leftarrow \Gamma(\vec{b}).P'\}\!\}, \Phi)$$
$$A = (\llbracket \mathcal{Q} \rrbracket_{\downarrow_{\mathsf{pi}}} \cup \{\!\{\llbracket \vec{x} \leftarrow \Gamma(\vec{b}).P' \rrbracket\}\!\}, \Phi)$$

  We call $(c_i)_i$ the private fresh names introduced by the translation $\llbracket \vec{x} \leftarrow \Gamma(\vec{b}).P' \rrbracket$, stressing that none of them appears in $\llbracket \mathcal{Q} \rrbracket_{\downarrow_{\mathsf{pi}}}$, $\llbracket P' \rrbracket$ nor Φ. Here $u \in \{c_i\}_i$ and we can therefore write:

$$A' = (\llbracket \mathcal{Q} \rrbracket_{\downarrow_{\mathsf{pi}}} \cup \{\!\{Q'\}\!\}, \Phi) \qquad \text{where } (\{\!\{\llbracket \vec{x} \leftarrow \Gamma(\vec{b}).P' \rrbracket\}\!\}, \Phi) \xrightarrow{\alpha}_{\mathsf{c}} (\{\!\{Q'\}\!\}, \Phi)$$

  If the sequence $\vec{b}$ contains a term which is not a message or does not reduce to a boolean, then one easily obtain that $(\{\!\{Q'\}\!\}, \Phi) \overset{\star}{\rightsquigarrow}_{\mathsf{pi}} (\mathcal{S}, \Phi)$ where $\mathcal{S}$ is silent in $(\llbracket \mathcal{Q} \rrbracket_{\downarrow_{\mathsf{pi}}}, \Phi)$ (for that we assume, w.l.o.g. that each input of Γ goes through at least one gate). Hence since $\{\!\{\vec{x} \leftarrow \Gamma(\vec{b}).P'\}\!\}$ is also silent in $(\mathcal{Q}, \Phi)$, we conclude with $B' = (\mathcal{Q}, \Phi)$.
  Otherwise assume that $Msg(\vec{b})$ and $\vec{b}\!\downarrow\, \subseteq \mathbb{B}$. Then one easily obtain by induction on the number of gates of Γ that $(\{\!\{Q'\}\!\}, \Phi) \overset{\star}{\rightsquigarrow}_{\mathsf{pi}} (\{\!\{\llbracket P' \rrbracket \{\vec{x} \mapsto \Gamma(\vec{b})\}\}\!\}, \Phi)$ and we conclude by choosing $B' = (\mathcal{Q} \cup \{\!\{P'\{\vec{x} \mapsto \Gamma(\vec{b})\}\}\!\}, \Phi)$.

· **case 5** (rules Choice-L, Choice-R, Choose-0, Choose-1 or Valuate):
The arguments of each of these cases are analogous to priorly-met subcases. ☐

In particular note that $A \overset{\star}{\rightsquigarrow} A_{\downarrow_{\mathsf{pi}}} \mathscr{R} \llbracket A_{\downarrow_{\mathsf{pi}}} \rrbracket_{\downarrow_{\mathsf{pi}}} = \llbracket A \rrbracket_{\downarrow_{\mathsf{pi}}} \overset{\star}{\leftsquigarrow} \llbracket A \rrbracket$ for all extended processes A, hence lemma 1.

A.3 Reductions in the pure π-calculus (section 4)

Proof of eq. (3) We prove first the correctness of our reduction for trace equivalence.

Proof.

($\Rightarrow$) Suppose that $A \not\approx_t B$. By a quick case analysis, we obtain a reduction $B \xrightarrow{\varepsilon}_{\mathsf{c}} (\{\!\{P(0)\}\!\}, \emptyset) \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\emptyset, \{\mathsf{ax}_1 \mapsto \mathsf{0}\})$ where $\mathsf{tr} = c(\vec{t}).\bar{c}\langle \mathsf{ax}_1 \rangle$ for some messages $\vec{t}$, such that for all reduction $A \overset{\mathsf{tr}}{\Rightarrow}_{\mathsf{c}} (\mathcal{C}, \Phi)$ the frames $\{\mathsf{ax}_1 \mapsto \mathsf{0}\}$ and Φ are not statically equivalent. In particular, for all $\vec{y} \subseteq \mathbb{B}$, by choosing $\Phi = \{\mathsf{ax}_1 \mapsto \varphi(\vec{t}, \vec{y})\}$ reachable from A, we obtain $\varphi(\vec{t}, \vec{y}) = \mathsf{0}$, hence the result.

($\Leftarrow$) Conversely, suppose that exists exists $\vec{x} \subseteq \mathbb{B}$ such that $\varphi(\vec{x}, \vec{y}) = \mathsf{1}$ for all $\vec{y} \subseteq \mathbb{B}$. Then the trace $B \overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} (\emptyset, \{\mathsf{ax}_1 \mapsto \mathsf{0}\})$ cannot be matched in A and therefore $A \not\approx_t B$.

☐

Proof of eq. (4) Now we prove the correctness of our reduction for observational equivalence, using the framework presented in appendix A.1.

Proof.

($\Rightarrow$) By contraposition, suppose that $\forall x_1 \exists y_1 \dots \forall x_n \exists y_n .\ \varphi(x_1, \dots, x_n, y_1, \dots, y_n) = 1$. For convinience we use a notation for subprocess extraction: is ℓ is a position of a process C, then the subprocess of C at position ℓ (which may not be closed) is denoted by $C_{|\ell}$. Then if:

$$C_i = \prod_{j=i+1}^{n+1} (\texttt{GetEnv}\,\langle A_j\rangle\,.A_j) \mid \prod_{j=i+1}^{n+1} (\texttt{GetEnv}\,\langle B_j\rangle\,.B_j)$$

we define $\mathscr{R}$ the smallest reflexive symmetric relation on closed extended processes such that:

1. $(A_i \mid C_i)(\overrightarrow{x}^{i-1}, \overrightarrow{y}^{i-1}) \mathscr{R} (B_i \mid C_i)(\overrightarrow{x}^{i-1}, \overrightarrow{y}^{i-1})$
   if $\forall x_i \exists y_i \dots \forall x_n \exists y_n, \varphi(\vec{x}, \vec{y})$.
2. $(A_{i|\ell} \mid C_i)(\overrightarrow{x}^{i}, \overrightarrow{y}^{i-1}) \mathscr{R} (B_{i|\ell} \mid C_i)(\overrightarrow{x}^{i-1}, \overrightarrow{y}^{i-1})$
   if $\ell \in \{0, 0.0\}$ and $\exists y_i \dots \forall x_n \exists y_n, \varphi(\vec{x}, \vec{y})$.
3. $(B_{i|0.0.1.\ell} \mid C_i)(\overrightarrow{x}^{i}, \overrightarrow{y}^{i}) \mathscr{R} (D_{i|0.\ell} \mid C_i)(\overrightarrow{x}^{i}, \overrightarrow{y}^{i})$
   if $\ell \in \{\varepsilon, 0\}$ and $\forall x_{i+1} \exists y_{i+1} \dots \forall x_n \exists y_n, \varphi(\vec{x}, \vec{y})$.

Then one can verify that $\mathscr{R}$ is a bisimulation up to $\rightsquigarrow$, and $A \mathscr{R} B$ by hypothesis, hence the $A \approx_\ell B$.

($\Leftarrow$) If we suppose that $\exists x_1 \forall y_1 \dots \exists x_n \forall y_n .\ \varphi(x_1, \dots, x_n, y_1, \dots, y_n) = 1$, then one can define a labelled attack $\mathscr{S}$ such that $B \mathscr{S} A$. We omit the concrete construction as it is analogous to that of $\mathscr{R}$ above; all in all this gives the conclusion $A \not\approx_\ell B$. □

A.4 Reductions in the applied π-calculus (section 5)

A.4.1 Static equivalence (theorem 3)

In this section we prove the correctness of the reduction presented in section 5.1. We refer to the notations of this section and write:

$$\Phi_1 = \{\mathsf{f}(0,k),\ \mathsf{f}(1,k)\} \qquad\qquad \Phi_2 = \{\mathsf{g}(0,k),\ \mathsf{g}(1,k)\}$$

We proceed by double implication.

Lemma 19. *If φ is satisfiable, $\Phi_1 \not\sim \Phi_2$.*

Proof. Let $\sigma = \{x_1 \mapsto b_1, \dots, x_n \mapsto t_n\}$ be a valuation satisfying φ. The booleans b_i are indistinctly seen as truth values or integers of $\{0,1\}$. Let us then define the recipe $\xi = \mathsf{eval}(\mathsf{ax}_{b_1+1}, \dots, \mathsf{ax}_{b_n+1})$. We have $Msg(\xi\Phi_1)$ but, since σ satisfies all clauses of φ, $\xi\Phi_2$ is irreducible. Since $root(\xi) \in \mathcal{F}_d$, we obtain $\neg Msg(\xi\Phi_2)$ and hence $\Phi_1 \not\sim \Phi_2$. □

Lemma 20. *If $\Phi_1 \not\sim \Phi_2$, φ is satisfiable.*

Proof. By non equivalence of Φ_1 and Φ_2, we know that there exist recipes ξ, ζ such that

H1 either $Msg(\xi\Phi_1)$ and $\neg Msg(\xi\Phi_2)$

H2 either $\neg Msg(\xi\Phi_1)$ and $Msg(\xi\Phi_2)$

H3 either $Msg(\xi\Phi_1)$, $Msg(\xi\Phi_2)$, $Msg(\zeta\Phi_1)$, $Msg(\zeta\Phi_2)$, $\xi\Phi_1{\downarrow} = \zeta\Phi_1{\downarrow}$ and $\xi\Phi_2{\downarrow} \neq \zeta\Phi_2{\downarrow}$,

H4 or $Msg(\xi\Phi_1)$, $Msg(\xi\Phi_2)$, $Msg(\zeta\Phi_1)$, $Msg(\zeta\Phi_2)$, $\xi\Phi_1{\downarrow} \neq \zeta\Phi_1{\downarrow}$ and $\xi\Phi_2{\downarrow} = \zeta\Phi_2{\downarrow}$.

Let ξ_0, ζ_0 be such recipes such that $|\xi_0| + |\zeta_0|$ is minimal. Let us then prove that φ is satisfiable. We proceed by case analysis on the hypothesis H1 to H4 satisfied by ξ_0, ζ_0.

- case 1: $Msg(\xi_0\Phi_1)$, $\neg Msg(\xi_0\Phi_2)$, and $root(\xi_0) \in \mathcal{F}_c \cup \mathcal{N}_{\mathsf{pub}} \cup \mathcal{AX}$

  It holds $root(\xi_0) \in \{\mathsf{f}, \mathsf{g}\}$ otherwise there would be an easy contradiction with hypothesis $\neg Msg(\xi_0\Phi_2)$. In particular, there exists a strict subterm of ξ_0 verifying H1 which contradicts the minimality of $|\xi_0| + |\zeta_0|$.

- case 2: $\neg Msg(\xi\Phi_1)$, $Msg(\xi\Phi_2)$, and $root(\xi_0) \in \mathcal{F}_c \cup \mathcal{N}_{\mathsf{pub}} \cup \mathcal{AX}$

  Symmetric to case 1.

- case 3: $Msg(\xi_0\Phi_1)$, $\neg Msg(\xi_0\Phi_2)$, and $root(\xi_0) = \mathsf{eval}$

  Hypothesis $Msg(\xi_0\Phi_1)$ yields

  $$\xi_0\Phi_1 = \mathsf{eval}(\mathsf{f}(t_1, t), \dots, \mathsf{f}(t_n, t)) \qquad \text{or} \qquad \xi_0\Phi_1 = \mathsf{eval}(\mathsf{g}(s_1, s), \dots, \mathsf{g}(s_n, s))$$

  for terms $t_1, \dots, t_n, t, s_1, \dots, s_n, s$ such that there is a clause C of φ with $vars(C) = \{x_{i_1}, x_{i_2}, x_{i_3}\}$, $Msg(s_{i_1})$, $Msg(s_{i_2})$, $Msg(s_{i_3})$, $\{s_{i_1}{\downarrow}, s_{i_2}{\downarrow}, s_{i_3}{\downarrow}\} \subseteq \{0, 1\}$ and the substitution $\{x_{i_1} \mapsto s_{i_1}{\downarrow}, x_{i_2} \mapsto s_{i_2}{\downarrow}, x_{i_3} \mapsto s_{i_3}{\downarrow}\}$ seen as a valuation of C falsifies C.

  - case 3a: $\xi_0\Phi_1 = \mathsf{eval}(\mathsf{f}(t_1, t), \dots, \mathsf{f}(t_n, t))$

    We write $\xi_0 = \mathsf{eval}(\xi_1, \dots, \xi_n)$. Several cases can arise:

    - Either $root(\xi_i) = \mathsf{f}$ for all $i \in \mathbb{N}_n$. Then there exists some $i \in \mathbb{N}_n$ and ξ a strict subterm of ξ_i such that $Msg(\xi\Phi_1)$ and $\neg Msg(\xi\Phi_2)$. In particular ξ verifies H1, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
    - Either $root(\xi_i) \in \mathcal{AX}$ for all $i \in \mathbb{N}_n$. Let us write $\xi_i = \mathsf{ax}_{b_i+1}$. In particular, hypothesis $\neg Msg(\xi_0\Phi_2)$ gives that $\mathsf{eval}(\mathsf{g}(b_1, k), \dots, \mathsf{g}(b_n, k))$ is irreducible. It entails there φ contains no clause falsified by the valuation $\{x_1 \mapsto b_1, \dots x_n \mapsto b_n\}$, hence φ is satisfiable.
    - Or $root(\xi_i) \in \mathcal{AX}$ and $root(\xi_j) = \mathsf{f}$ for some $i, j \in \mathbb{N}_n$. We observe that k is not deducible from Φ_1 (i.e. there exists no recipe ζ such that $Msg(\zeta)$ and $\zeta\Phi_1{\downarrow} = k$). In particular $\neg Msg(\xi_0\Phi_1)$, hence a contradiction.

  - case 3b: $\xi_0\Phi_1 = \mathsf{eval}(\mathsf{g}(s_1, s), \dots, \mathsf{g}(s_n, s))$

    We derive a contradiction. Indeed it implies $\xi_0 = \mathsf{eval}(\mathsf{g}(\xi_1, \xi_1'), \dots, \mathsf{g}(\xi_n, \xi_n'))$ for recipes $\xi_1, \dots, \xi_n$, $\xi_1', \dots, \xi_n'$. Then hypothesis $\neg Msg(\xi_0\Phi_2)$ yields different cases:

    - Either there exists $i \in \mathbb{N}_n$ such that $\neg Msg(\xi_i\Phi_2)$. Then ξ_i satisfies H1, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.

  - Either there exists $i \in \mathbb{N}_n$ such that $\neg Msg(\xi'_i\Phi_2)$. Then ξ'_i satisfies H1, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
  - Either there exist $i, j \in \mathbb{N}_n$ such that $Msg(\xi'_i\Phi_2)$, $Msg(\xi'_j\Phi_2)$ and $\xi'_i\Phi_2\!\downarrow \neq \xi'_j\Phi_2\!\downarrow$. Then ξ'_i, ξ'_j would satisfy H3, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
  - Or there exist $i \in \{i_1, i_2, i_3\}$ such that $Msg(\xi_i\Phi_2)$ and $\xi_i\Phi_1\!\downarrow \neq \xi_i\Phi_2\!\downarrow$ (otherwise that would contradict $\neg Msg(\xi_0\Phi_2)$). Since $\xi_i\Phi_1\!\downarrow = s_i \in \{\mathsf{0}, \mathsf{1}\}$, there is a recipe $\zeta \in \{\mathsf{0}, \mathsf{1}\}$ such that $\xi_i\Phi_1\!\downarrow = \zeta\Phi_1\!\downarrow$ and $\xi_i\Phi_2\!\downarrow \neq \zeta\Phi_2\!\downarrow$. In particular ξ_i, ζ satisfy H3, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.

- case 4: $\neg Msg(\xi_0\Phi_1)$, $Msg(\xi_0\Phi_2)$, and $root(\xi_0) = \mathsf{eval}$

  Hypothesis $Msg(\xi_0\Phi_2)$ yields

  $$\xi_0\Phi_2 = \mathsf{eval}(\mathsf{f}(t_1, t), \dots, \mathsf{f}(t_n, t)) \qquad \text{or} \qquad \xi_0\Phi_2 = \mathsf{eval}(\mathsf{g}(s_1, s), \dots, \mathsf{g}(s_n, s))$$

  for terms $t_1, \dots, t_n, t, s_1, \dots, s_n, s$ such that there is a clause C of φ with $vars(C) = \{x_{i_1}, x_{i_2}, x_{i_3}\}$, $Msg(s_{i_1})$, $Msg(s_{i_2})$, $Msg(s_{i_3})$, $\{s_{i_1}\!\downarrow, s_{i_2}\!\downarrow, s_{i_3}\!\downarrow\} \subseteq \{\mathsf{0}, \mathsf{1}\}$ and the substitution $\{x_{i_1} \mapsto s_{i_1}\!\downarrow, x_{i_2} \mapsto s_{i_2}\!\downarrow, x_{i_3} \mapsto s_{i_3}\!\downarrow\}$ seen as a valuation of C falsifies C.

  - case 4a: $\xi_0\Phi_2 = \mathsf{eval}(\mathsf{f}(t_1, t), \dots, \mathsf{f}(t_n, t))$

    We derive a contradiction. Indeed it implies $\xi_0 = \mathsf{eval}(\mathsf{f}(\xi_1, \xi'_1), \dots, \mathsf{f}(\xi_n, \xi'_n))$ for recipes $\xi_1, \dots, \xi_n$, $\xi'_1, \dots, \xi'_n$. Then hypothesis $\neg Msg(\xi_0\Phi_1)$ yields different cases:
    - Either there exists $i \in \mathbb{N}_n$ such that $\neg Msg(\xi_i\Phi_1)$. Then ξ_i satisfies H2, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
    - Either there exists $i \in \mathbb{N}_n$ such that $\neg Msg(\xi'_i\Phi_1)$. Then ξ'_i satisfies H2, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
    - Or there exist $i, j \in \mathbb{N}_n$ such that $Msg(\xi'_i\Phi_1)$, $Msg(\xi'_j\Phi_1)$ and $\xi'_i\Phi_1\!\downarrow \neq \xi'_j\Phi_1\!\downarrow$. Then ξ'_i, ξ'_j would satisfy H4, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
  - case 4b: $\xi_0\Phi_2 = \mathsf{eval}(\mathsf{g}(s_1, s), \dots, \mathsf{g}(s_n, s))$

    We write $\xi_0 = \mathsf{eval}(\xi_1, \dots, \xi_n)$. Several cases can arise:
    - Either $root(\xi_i) = \mathsf{g}$ for all $i \in \mathbb{N}_n$. Then there exists some $i \in \mathbb{N}_n$ and ξ a strict subterm of ξ_i such that $Msg(\xi\Phi_2)$ and $\neg Msg(\xi\Phi_1)$. In particular ξ verifies H2, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
    - Either $root(\xi_i) \in \mathcal{AX}$ for all $i \in \mathbb{N}_n$. There is a clear contradiction with $\neg Msg(\xi_0\Phi_1)$.
    - Or $root(\xi_i) \in \mathcal{AX}$ and $root(\xi_j) = \mathsf{g}$ for some $i, j \in \mathbb{N}_n$. We observe that k is not deducible from Φ_2 (i.e. there exists no recipe ζ such that $Msg(\zeta)$ and $\zeta\Phi_2\!\downarrow = k$). In particular $\neg Msg(\xi_0\Phi_2)$, hence a contradiction.

- case 5: $Msg(\xi_0\Phi_1)$, $Msg(\xi_0\Phi_2)$, $Msg(\zeta_0\Phi_1)$, $Msg(\zeta_0\Phi_2)$, $\xi_0\Phi_1\!\downarrow = \zeta_0\Phi_1\!\downarrow$, $\xi_0\Phi_2\!\downarrow \neq \zeta_0\Phi_2\!\downarrow$

  We show that this case cannot arise: the fact that $\xi_0\Phi_1$, $\xi_0\Phi_2$, $\zeta_0\Phi_1$ and $\zeta_0\Phi_2$ are messages will always contradict either $\xi_0\Phi_1\!\downarrow = \zeta_0\Phi_1\!\downarrow$ or $\xi_0\Phi_2\!\downarrow \neq \zeta_0\Phi_2\!\downarrow$.

  - case 5a: $\{root(\xi_0), root(\zeta_0)\} \subseteq \{\mathsf{0}, \mathsf{1}\} \cup \mathcal{N}_{\mathsf{pub}} \cup \mathcal{AX}$

    Then we can always derive a contradiction with $\xi_0\Phi_1\!\downarrow = \zeta_0\Phi_1\!\downarrow$ and $\xi_0\Phi_2\!\downarrow \neq \zeta_0\Phi_2\!\downarrow$.

- case 5b: $\{root(\xi_0), root(\zeta_0)\} \subseteq \{\mathsf{f}, \mathsf{g}\}$
  If $root(\xi_0) \neq root(\zeta_0)$, this yields a contradiction with $\xi_0\Phi_1{\downarrow} = \zeta_0\Phi_1{\downarrow}$. Otherwise we can find strict subterms of ξ_0 and ζ_0 satisfying H3, yielding a contradiction with minimality of $|\xi_0| + |\zeta_0|$.
- case 5c: $root(\xi_0) \in \{\mathsf{f}, \mathsf{g}\}$ and $root(\zeta_0) \in \{0, 1\} \cup \mathcal{N}_{\mathsf{pub}}$ (or conversely)
  This yields a contradiction with $\xi_0\Phi_1{\downarrow} = \zeta_0\Phi_1{\downarrow}$.
- case 5d: $root(\xi_0) = \mathsf{f}$ and $root(\zeta_0) \in \mathcal{AX}$ (or conversely)
  Since k is not deducible from Φ_1, this yields a contradiction with $\xi_0\Phi_1{\downarrow} = \zeta_0\Phi_1{\downarrow}$.
- case 5e: $root(\xi_0) = \mathsf{g}$ and $root(\zeta_0) \in \mathcal{AX}$ (or conversely)
  This yields a clear contradiction with $\xi_0\Phi_1{\downarrow} = \zeta_0\Phi_1{\downarrow}$.
- case 5f: $root(\xi_0) = \mathsf{eval}$ and $root(\zeta_0) \notin \{\mathsf{eval}, 0\}$ (or conversely)
  This yields a contradiction with $\xi_0\Phi_1{\downarrow} = \zeta_0\Phi_1{\downarrow}$, since $root(\xi_0\Phi_1{\downarrow}) \neq root(\zeta_0\Phi_1{\downarrow})$.
- case 5g: $\{root(\xi_0), root(\zeta_0)\} \subseteq \{0, \mathsf{eval}\}$
  Since $Msg(\xi_0\Phi_2)$ and $Msg(\zeta_0\Phi_2)$, this yields $\xi_0\Phi_2{\downarrow} = 0 = \zeta_0\Phi_2{\downarrow}$, in contradiction with hypothesis $\xi_0\Phi_2{\downarrow} \neq \xi_0\Phi_2{\downarrow}$.

- case 6: $Msg(\xi_0\Phi_1)$, $Msg(\xi_0\Phi_2)$, $Msg(\zeta_0\Phi_1)$, $Msg(\zeta_0\Phi_2)$, $\xi_0\Phi_1{\downarrow} \neq \zeta_0\Phi_1{\downarrow}$, $\xi_0\Phi_2{\downarrow} = \zeta_0\Phi_2{\downarrow}$
  A contradiction can be derived symmetrically to case 5. □

A.4.2 Trace and observational equivalences (lemmas 3 to 5 section 5.2)

Before concretly proving the pending lemmas, let us introduce some notations and prove intermediary results about static equivalence. Sticking to the notations of section 5.2 (in particular we fix a private nonce $s \in \mathcal{N}_{\mathsf{prv}}$), we define the following frames given a protocol term t:

$$\Phi_t = \{\mathsf{ax}_1 \mapsto \mathsf{h}(t, s),\ \mathsf{ax}_2 \mapsto \mathsf{h}(1, s)\}$$
$$\Phi_t^{\mathsf{N}} = \{\mathsf{ax}_1 \mapsto \mathsf{h}_{\mathsf{N}}(t, s),\ \mathsf{ax}_2 \mapsto \mathsf{h}(1, s)\}$$
$$\Phi_t^{\mathbb{B}} = \{\mathsf{ax}_1 \mapsto \mathsf{h}_{\mathbb{B}}(t, s),\ \mathsf{ax}_2 \mapsto \mathsf{h}(1, s)\}$$

One shall observe that the whole deal with our reduction is about which instances of these three frames are reachable in which conditions. Hence first we prove a lemma investigating the static equivalence between some of them:

Lemma 21. *Let t be a message in normal form ($t = t{\downarrow}$). The following properties hold:*

(i) $\Phi_t \sim \Phi_0$ **iff** $t \neq 1$

(ii) $\Phi_t^{\mathsf{N}} \sim \Phi_0$ **iff** $root(t) \neq \mathsf{Node}$

(iii) $\Phi_t^{\mathbb{B}} \sim \Phi_0$ **iff** $t \notin \mathbb{B}$

Proof. We prove the three equivalences together by double implication.

($\Rightarrow$) We prove the three properties by contraposition. We naturally proceed by exhibiting ground recipes ξ, ζ witnessing the non-static-equivalence goal:

· *(i)* We assume $t = \mathsf{1}$ and we choose $\xi = \mathsf{ax}_1$ and $\zeta = \mathsf{ax}_2$: the conclusion follows from $\xi\Phi_t{\downarrow} = \mathsf{h}(\mathsf{1}, s) = \zeta\Phi_t{\downarrow}$ and $\xi\Phi_{\mathsf{0}}{\downarrow} = \mathsf{h}(\mathsf{0}, s) \neq \mathsf{h}(\mathsf{1}, s) = \zeta\Phi_{\mathsf{0}}{\downarrow}$.

· *(ii)* We assume $t = \mathsf{Node}(t_1, t_2)$ and we choose $\xi = \mathsf{TestNode}(\mathsf{ax}_1)$: the conclusion follows from $\mathit{Msg}(\xi\Phi_t^{\mathsf{N}})$ and $\neg\mathit{Msg}(\xi\Phi_{\mathsf{0}}^{\mathsf{N}})$.

· *(iii)* We assume $t \in \{\mathsf{0}, \mathsf{1}\}$ and we choose $\xi = \mathsf{TestBool}(\mathsf{ax}_1)$: the conclusion follows from $\mathit{Msg}(\xi\Phi_t^{\mathbb{B}})$ and $\neg\mathit{Msg}(\xi\Phi_{\mathsf{0}}^{\mathbb{B}})$.

($\Leftarrow$) The key point is an observation about rewriting critical pairs (not specific to $\mathcal{R}$):

if: u is a term;

if: σ is a substitution such that for any $m \in \mathit{img}(\sigma)$ there exists no rule $\ell \to r$ of $\mathcal{R}$ such that m is unifiable with a subterm $u \in \mathit{st}(\ell) - \mathcal{X}$;

then: for any rewriting sequence $u\sigma \to^\star s$, it holds that $s = u'\sigma$ for some $u \to^\star u'$ (where u' is in normal form **iff** s is in normal form. In particular $(u\sigma){\downarrow} = (u{\downarrow})\sigma$).

One shall note that any frame Φ investigated by the lemma (Φ_t when $t \neq \mathsf{1}$, Φ_t^{N} when $\mathit{root}(t) \neq \mathsf{Node}$ and $\Phi_t^{\mathbb{B}}$ when $t \notin \mathbb{B}$) verifies the second hypothesis. As a consequence, if ξ is a ground recipe such that $\mathit{axioms}(\xi) \subseteq \mathit{dom}(\Phi)$, then $\mathit{Msg}(\xi\Phi)$ **iff** $\mathit{Msg}(\xi\Phi_{\mathsf{0}})$ (**iff** $\forall\zeta \in \mathit{st}(\xi),\ \zeta{\downarrow} \in \mathcal{T}(\mathcal{F}_c, \mathcal{N}_{\mathsf{pub}} \cup \mathcal{AX})$). This settles the first item of the definition of static equivalence. As for the second item, let us fix two ground recipes ξ, ζ such that $\mathit{Msg}(\xi\Phi)$ and $\mathit{Msg}(\xi\Phi_{\mathsf{0}})$. Let us then prove that $\xi\Phi{\downarrow} = \zeta\Phi{\downarrow}$ **iff** $\xi\Phi_{\mathsf{0}}{\downarrow} = \zeta\Phi_{\mathsf{0}}{\downarrow}$ by induction on $(\xi{\downarrow}, \zeta{\downarrow})$. Note that we will intensively (and implicitly) use the fact that $\xi\Phi{\downarrow} = (\xi{\downarrow})\Phi$ (same for ζ and/or Φ_{0}).

· **case 1** : $\xi{\downarrow} = f(\xi_1, \ldots, \xi_n)$ and $\zeta{\downarrow} = f(\zeta_1, \ldots, \zeta_p)$ with $f, g \in \mathcal{F}_c$.

If $f = g$ then the result follows from induction hypothesis and if $f \neq g$ the conclusion is immediate ($\xi\Phi{\downarrow} \neq \zeta\Phi{\downarrow}$ and $\xi\Phi_{\mathsf{0}}{\downarrow} \neq \zeta\Phi_{\mathsf{0}}{\downarrow}$).

· **case 2** : $\xi{\downarrow} \in \mathcal{AX}$ and $\zeta{\downarrow} \in \mathcal{AX}$.

If $\xi = \zeta$ the conclusion is immediate and so is it when $\xi \neq \zeta$ since $\Phi(\mathsf{ax}_1) \neq \Phi(\mathsf{ax}_2)$ and $\Phi_{\mathsf{0}}(\mathsf{ax}_1) \neq \Phi_{\mathsf{0}}(\mathsf{ax}_2)$.

· **case 3** : $\xi{\downarrow} \in \mathcal{AX}$ and $\zeta{\downarrow} = f(\zeta_1, \ldots, \zeta_p)$ with $f \in \mathcal{F}_c$.

We argue that $\xi\Phi{\downarrow} \neq \zeta\Phi{\downarrow}$ and $\xi\Phi_{\mathsf{0}}{\downarrow} \neq \zeta\Phi_{\mathsf{0}}{\downarrow}$. Either of the two equalities being verified would indeed imply that $s \in \{\zeta_2\Phi, \zeta_2\Phi_{\mathsf{0}}\}$: this is impossible as ζ_2 is a ground recipe in normal form and $s \in \mathcal{N}_{\mathsf{prv}}$ (one easily shows that $\zeta_2\Phi$ and $\zeta_2\Phi_{\mathsf{0}}$ are either public names, constants, or termes of height 1 or more).

As $\mathit{Msg}(\xi\Phi)$ and $\mathit{Msg}(\xi\Phi_{\mathsf{0}})$, the preliminary observation justifies that $\xi{\downarrow}$ and $\zeta{\downarrow}$ function symbols are all constructor. In particular no other cases than the three above need to be considered, which concludes the proof. $\square$

With this lemma in mind, the proofs of lemmas 3 and 4 become quite straightforward:

Lemma 3. *Let x be a message which is not complete binary tree of height n with boolean leaves. Then there exists a reduction* $\mathtt{CheckTree}(x) \overset{\xi}{\Rightarrow}_{\mathsf{c}} (\{\!\{P\}\!\}, \emptyset)$ *such that* $(\{\!\{P\}\!\}, \emptyset) \approx_\ell (\overline{c}\langle \mathsf{h}(\mathsf{0}, s)\rangle.\, \overline{c}\langle \mathsf{h}(\mathsf{1}, s)\rangle, \emptyset)$.

Proof. Let x be a message which is not a complete binary tree of height n whose leaves are booleans.

- **case 1** : there exists a position $\vec{\pi} \in \mathbb{B}^\star$ such that $|\vec{\pi}| = i \in [\![0, n-1]\!]$ and $root(x_{|\vec{\pi}}) \neq \mathsf{Node}$.
  The result follows from lemma 21 after writing the following sequence of transitions:

$$\begin{aligned}\texttt{CheckTree}(x) &\overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} \texttt{Choose}(p_1, \ldots, p_i).\ \overline{c}\langle \mathsf{h_N}(x_{|p_1\cdots p_i}, s)\rangle.\ \overline{c}\langle \mathsf{h}(1, s)\rangle \\ &\overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} \overline{c}\langle \mathsf{h_N}(x_{|\vec{\pi}}, s)\rangle.\ \overline{c}\langle \mathsf{h}(1, s)\rangle\end{aligned}$$

- **case 2** : there exists a position $\vec{\pi} \in \mathbb{B}^n$ such that $x_{|\vec{\pi}} \notin \mathbb{B}$.
  The result follows from lemma 21 after writing the following sequence of transitions:

$$\begin{aligned}\texttt{CheckTree}(x) &\overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} \texttt{Choose}(p_1, \ldots, p_n).\ \overline{c}\langle \mathsf{h_N}(x_{|p_1\cdots p_n}, s)\rangle.\ \overline{c}\langle \mathsf{h}(1, s)\rangle \\ &\overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} \overline{c}\langle \mathsf{h}_{\mathbb{B}}(x_{|\vec{\pi}}, s)\rangle.\ \overline{c}\langle \mathsf{h}(1, s)\rangle\end{aligned}$$ □

Lemma 4. *Let x be a complete binary tree of height n whose leaves are booleans, and val_x be the valuation mapping the variable number $i = \sum_{k=1}^m p_k 2^{k-1}$ of $[\![\Gamma]\!]_\varphi$ to $x_{|p_1\cdots p_m} \in \mathbb{B}$. Then, if val_x does not satisfy $[\![\Gamma]\!]_\varphi$ then there exists $\texttt{CheckSat}(x) \overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} (\{\!\{P\}\!\}, \emptyset)$ such that $(\{\!\{P\}\!\}, \emptyset) \approx_\ell (\overline{c}\langle \mathsf{h}(0, s)\rangle. \overline{c}\langle \mathsf{h}(1, s)\rangle, \emptyset)$.*

Proof. Let x be a complete binary tree of height n whose leaves are booleans, that is to say, a message such that $x_{|\vec{p}} \in \mathbb{B}$ for all $\vec{p} \in \mathbb{B}^n$. Naming $x_0, \ldots, x_{2^n-1}$ the variables of $[\![\Gamma]\!]_\varphi$ in this order, val_x refers to the valuation mapping x_i to $x_{|\vec{p}}$ where $\vec{p}$ is the binary representation of i (of size n with padding head 0's).

Let us now assume that val_x does not satisfy $[\![\Gamma]\!]_\varphi$. In particular there exists a clause of $[\![\Gamma]\!]_\varphi$, say the i^{th} clause with $i = \sum_{k=1}^m \pi_k 2^{k-1}$, which is falsified by val_x. In particular, if the three variable of this clause are called x_1, x_2, x_3 with respective negation bits b_1, b_2, b_3, the following formula is evaluated to false (i.e. $\mathsf{0}$):

$$\bigvee_{i=1}^{3} (\ b_j = \mathsf{val}_x(x_j)\ )$$

Therefore, by choosing the sequence $\pi_1, \ldots, \pi_m$ to instanciate the initial $\texttt{Choose}(p_1, \ldots, p_m)$ of $\texttt{CheckSat}(x)$, we obtain the following sequence of transitions, which concludes the proof:

$$\texttt{CheckSat}(x) \overset{\varepsilon}{\Rightarrow}_{\mathsf{c}} \overline{c}\langle \mathsf{h}(0, s)\rangle.\ \overline{c}\langle \mathsf{h}(1, s)\rangle$$ □

We finally gathered all the ingredients needed to prove the main lemma:

Lemma 5. $[\![\Gamma]\!]_\varphi$ *is satisfiable* **iff** $A \not\approx_t B$ **iff** $A \not\approx_\ell B$.

Proof. We name the three properties as follows: (i) $[\![\Gamma]\!]_\varphi$ is satisfiable, (ii) $A \not\approx_t B$ and (iii) $A \not\approx_o B$. We reach the conclusion by the following chain of implications.
$(i) \Rightarrow (ii)$: Let us consider a valuation satisfying φ and let t be a message such that val_t is equal to this valuation. Then since the trace $B \xRightarrow{c(t).\overline{c}\langle \mathsf{ax}_1\rangle.\overline{c}\langle \mathsf{ax}_2\rangle}_{\mathsf{c}} (\emptyset, \Phi_0)$ cannot be matched

in A by lemma 21, we obtain $B \not\sqsubseteq_t A$. Hence the (ii).
$(ii) \Rightarrow (iii)$: Follows from the inclusion of relations $\approx_o \subseteq \approx_t$.
$\neg(i) \Rightarrow \neg(iii)$: Let us consider $\mathscr{R}$ the smallest reflexive symmetric relation on extended processes such that:

1. $A \mathscr{R} B$
2. $A'(x) \mathscr{R} B'(x)$ for all message x, where $A = c(x).A'(x)$ and $B = c(x).B'(x)$
3. $P_i \mathscr{R} P'_i$, where:

$$\begin{aligned} P_i &= (\{\!\!\{\overline{c}\langle t_{i+1}\rangle \dots \overline{c}\langle t_p\rangle\}\!\!\}, \{\mathsf{ax}_1 \mapsto t_1, \dots, \mathsf{ax}_i \mapsto t_i\}) \\ P'_i &= (\{\!\!\{\overline{c}\langle t'_{i+1}\rangle \dots \overline{c}\langle t'_p\rangle\}\!\!\}, \{\mathsf{ax}_1 \mapsto t'_1, \dots, \mathsf{ax}_i \mapsto t'_i\}) \end{aligned}$$

and where $\{\mathsf{ax}_1 \mapsto t_1, \dots, \mathsf{ax}_p \mapsto t_p\} \sim \{\mathsf{ax}_1 \mapsto t'_1, \dots, \mathsf{ax}_p \mapsto t'_p\}$

It easily follows from lemmas 3, 4 and 21 that $\mathscr{R}$ is a bisimulation up to $\rightsquigarrow$, hence the conclusion. $\square$

B Proofs of for the partition trees

B.1 Invariants on the sets of extended symbolic processes

We first provide some invariants of on the sets of extended symbolic processes that will always be satisfied during the course of the procedure.

Definition 34 (Well-formed). *We define the predicate $\mathcal{P}_{form}$ defined on extended constraint systems such that for all constraint systems $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$, $\mathcal{P}_{form}(\mathcal{C})$ holds when*

- *Variables in K and F : $vars^2(\mathsf{K}, \mathsf{F}) \subseteq vars^2(\mathsf{D})$*
- *Variables in E^1: $dom(mgu(\mathsf{E}^1|_=)) \cap vars^1(\mathsf{D}) = \emptyset$, $vars^1(img(mgu(\mathsf{E}^1|_=))) \subseteq vars^1(\mathsf{D})$.*
- *Variables in E^2: $vars^2(img(mgu(\mathsf{E}^2|_=))) \subseteq vars^2(\mathsf{D})$ and for all $\xi \in img(mgu(\mathsf{E}^2|_=))$, $\xi \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$.*
- *Shape of K: for all $(\xi \vdash^? u) \in \mathsf{K}$, $u \notin \mathcal{X}^1$, $u \in st(\Phi)$ and for all $\xi' \in st(\xi)$, $\xi' \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$.*
- *Shape of F: For all $\forall S.H \Leftarrow \varphi \in \mathsf{F}(\mathcal{C})$, φ only contains syntactic equations as hypothesis, i.e. no deduction fact; moreover, for all $\xi \in sst(H)$, $\xi \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$.*

Definition 35 (Soundness). *We define the predicate $\mathcal{P}_{sound}$ defined on extended constraint systems such that for all constraint systems $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$, $\mathcal{P}_{sound}(\mathcal{C})$ holds when for all $\psi \in \mathsf{K} \cup \mathsf{F}$, for all substitutions Σ, σ, if $D = \{\psi' \in \mathsf{D} \mid vars^2(\psi') \subseteq vars^2(\psi)\}$ and $(\Phi\sigma, \Sigma, \sigma) \models \mathsf{E}^1|_= \wedge D$ then $(\Phi\sigma, \Sigma, \sigma) \models \psi$.*

The soundness property states that if the free variables of a deduction or equality formula ψ are correctly instantiated by Σ and σ w.r.t. to D then $\psi\Sigma\sigma$ is a tautology. In other word, $(\Phi\sigma, \Sigma, \sigma)$ models ψ.

Definition 36 (Completeness). *We define the predicate $\mathcal{P}_{comp}$ defined on extended constraint systems such that for all constraint systems $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$, $\mathcal{P}_{comp}(\mathcal{C})$ if and only if for all $\psi = (\forall S.H \Leftarrow \varphi) \in \mathsf{F}$, for all substitutions Σ, σ, if $(\Phi\sigma, \Sigma, \sigma) \models \mathsf{D} \wedge \mathsf{E}^1|_= \wedge \mathsf{E}^2$ and $(\Sigma, sst^2(\mathsf{K}\Sigma), sst(H\Sigma))$ is uniform in $(\Phi\sigma, \mathsf{K}\Sigma\sigma)$ then*

- *if $H = \xi \vdash^? u$ and $Msg(\xi\Sigma\Phi\sigma)$ then there exist $\psi' = (\forall S'.\xi' \vdash^? u' \Leftarrow \varphi') \in \mathsf{F}$ and σ' such that $dom(\sigma') = vars^1(S')$, $\psi \simeq_{\mathsf{r}} \psi'$ and $(\Phi\sigma, \Sigma, \sigma\sigma') \models \varphi'$.*
- *if $H = \xi_1 =_f^? \xi_2$ and $\Phi\sigma \models \xi_1\Sigma =_f^? \xi_2\Sigma$ then there exist $\psi' = (\forall S'.\xi_1' =_f^? \xi_2' \Leftarrow \varphi') \in \mathsf{F}$ and a substitution σ' such that $dom(\sigma') = vars^1(S')$, $\psi \simeq_{\mathsf{r}} \psi'$, $(\Phi\sigma, \Sigma, \sigma) \models \varphi'$.*

For a deduction or equality formula $\forall S.H \Leftarrow \varphi$, the soundness predicate indicate that if φ is satisfied then so is H. Intuitively, with the completeness predicate, we indicate that if some instantiation of H can be satisfied then there exists some other formula $\forall S'.H' \Leftarrow \varphi'$ in F where the same instantiation satisfies φ' (and also satisfies H' by soundness). In other words, the completeness predicate ensures that we did not forget any cases.

All three previous invariants are lifted to set of extended symbolic processes in the sense that all extended constraint in them should satisfy them.

When we apply rules with parameter k, we are typically saturating the set of solved deduction facts K until every messages deducible with a recipe in $\mathcal{T}(\mathcal{F}, \mathcal{AX}_k \cup \mathcal{N}_{\mathsf{pub}})$ is a consequence of K restricted to the deduction formulas having universal second-order variables in $\mathcal{X}_k^2 \smallsetminus \mathcal{X}_{k-1}^2$.

Definition 37. *We define the predicate $\mathcal{P}_{cons}$ defined on set of extended symbolic process S such that $\mathcal{P}_{cons}(S)$ holds if and only if for all $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e) \in S$, by considering the minimal k such that $vars^2(\mathsf{D}(\mathcal{C}^e)) \subseteq \mathcal{X}_k^2$, we have that the following holds:*

For all $\mathsf{f}/n \in \mathcal{F}_d$, for all $(\xi_1, u_1), \dots, (\xi_n, u_n) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}))$ such that $\xi_1, \dots, \xi_n \in \mathcal{T}_k^2$, if $\mathsf{f}(u_1, \dots, u_n)\!\downarrow$ is a constructor protocol term then there exists $\xi \in \mathcal{T}_k^2$ such that $(\xi, \mathsf{f}(u_1, \dots, u_n)\!\downarrow) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}) \cup \mathsf{D}(\mathcal{C}))$.

Note that in the previous definition, we only the deduction fact from D with second-order variable in $\mathcal{X}_{k-1}^2$ to be satisfied.

Definition 38. *We define the predicate $\mathcal{P}_{set}$ defined on set S of extended symbolic processes S such that $\mathcal{P}_{vect}(S)$ holds if and only if for all $(\mathcal{P}_1, \mathcal{C}_1, \mathcal{C}_1^e), (\mathcal{P}_2, \mathcal{C}_2, \mathcal{C}_2^e) \in S$, for all $\psi_1 \in \mathsf{F}(\mathcal{C}_1^e)$, if for all $\psi_2 \in \mathsf{F}(\mathcal{C}_2^e)$, $\psi_1 \not\simeq_{\mathsf{r}} \psi_2$ then for all $(\Sigma, \sigma) \in Sol_e(\mathcal{C}_2)$,*

- *if $\psi_1 = (\forall S.\xi \vdash^? u \Leftarrow \varphi)$ then $(\Phi(\mathcal{C}_2)\sigma, \Sigma) \models \xi \neq_f^? \xi$*
- *if $\psi_2 = (\forall S.\xi_1 =_f^? \xi_2 \Leftarrow \varphi)$ then $(\Phi(\mathcal{C}_2)\sigma, \Sigma) \models \xi_1 \neq_f^? \xi_2$*

When we apply a case distinction rule on a set of extended symbolic processes, we in fact apply the same rule with same parameters on each extended constraint systems of the set. Thus, if a deduction formula $\forall S.\xi \vdash^? u \Leftarrow \varphi$ is present in a constraint system $\mathcal{C}$, it intuitively means that we guessed how an instantiation of ξ could produce a message, that is instantiation of u. Thus, since we apply the same rules on all the other constraint systems, similar guesses, *i.e.* recipe equivalent formulas, should appear in all the other constraint systems. Therefore, when such recipe equivalent formulas cannot be found in a constraint system $\mathcal{C}_2$, it ensures us that no instantiation of ξ can produce a message in $\mathcal{C}_2$.

Definition 39. *We define the predicate $\mathcal{P}_{sol}$ defined on extended symbolic processes $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e)$ such that $\mathcal{P}_{sol}((\mathcal{P}, \mathcal{C}, \mathcal{C}^e))$ holds if and only if*

- *$vars^2(\mathcal{C}) \subseteq \mathcal{X}^2_i$ iff $vars^2(\mathcal{C}^e) \subseteq \mathcal{X}^2_i$, for all i.*
- *for all $(\Sigma, \sigma) \in Sol_e(\mathcal{C}^e)$, $(\Sigma|_{vars^2(\mathcal{C})}, \sigma|_{vars^1(\mathcal{C})} \in Sol(\mathcal{C})$.*

This invariant indicates that all solutions of the extended constraint systems $\mathcal{C}^e$ is also solutions of the symbolic constraint system $\mathcal{C}$. We restrict the substitutions to the variables of $\mathcal{C}$ since our extended constraint system may introduce new variables (e.g. by applying most general solutions) but all these variables are uniquely defined by the instantiation of the variables of $\mathcal{C}$. As usual, we lift this invariant to set of (set of) extended symbolic processes.

For readability, we will denoted $\mathcal{P}_{all}(S)$ the predicate to hold when $\mathcal{P}_{vect}(S), \mathcal{P}_{form}(S)$, $\mathcal{P}_{comp}(S), \mathcal{P}_{sound}(S)$, $\mathcal{P}_{cons}(S)$ and $\mathcal{P}_{sol}(S)$ hold. Moreover, we will also lift these predicates to sets of sets of extended symbolic processes.

B.2 Preservation by application of substitutions

In this section, we show that applying substitution preserves in some cases the different notions we use in our algorithms, namely first-order and second-order equations, deduction and equality facts; and uniformity.

Lemma 22. *Let ψ be either a deduction fact, or an equality fact or a first-order equation or a second-order equation. For all ground frame Φ, for all substitutions $\Sigma, \Sigma', \sigma, \sigma'$ if $dom(\Sigma) \cap dom(\Sigma') = \emptyset$ and $dom(\sigma) \cap dom(\sigma') = \emptyset$ then $(\Phi, \Sigma\Sigma', \sigma\sigma') \models \psi$ is equivalent to $(\Phi, \Sigma\Sigma', \sigma\sigma') \models \psi\Sigma\sigma$ and is equivalent to $(\Phi, \Sigma', \sigma') \models \psi\Sigma\sigma$.*

Proof. The proof of this lemma is done by case analysis on ψ.

Case $u =^? v$: Consider $(\Phi, \Sigma\Sigma', \sigma\sigma') \models u =^? v$. This is equivalent to $u\sigma\sigma' = v\sigma\sigma'$. Since $vars(\Sigma) \cap \mathcal{X}^1 = \emptyset$, we deduce that $(\Phi, \Sigma\Sigma', \sigma\sigma') \models u =^? v$ is equivalent to $u\Sigma\sigma\sigma' = v\Sigma\sigma\sigma'$. This is also equivalent to $u\Sigma\sigma\sigma\sigma' = v\Sigma\sigma\sigma\sigma'$ and so $(\Phi, \Sigma\Sigma', \sigma\sigma') \models u\Sigma\sigma =^? v\Sigma\sigma$. Note that $u\Sigma\sigma\sigma' = v\Sigma\sigma\sigma'$ is also equivalent to $(\Phi, \Sigma', \sigma') \models u\Sigma\sigma =^? v\Sigma\sigma$.

Case $\xi =^? \xi'$: Similar to the previous case.

Case $\xi \vdash^? u$: Consider $(\Phi, \Sigma\Sigma', \sigma\sigma') \models \xi \vdash^? u$. It is equivalent to $\xi\Sigma\Sigma'\Phi{\downarrow} = u\sigma\sigma'$ and $Msg(\xi\Sigma\Sigma'\Phi)$. But $(\xi \vdash^? u)\Sigma\sigma = \xi\Sigma \vdash^? u\sigma$. Moreover, by definition of substitution (in particular the acyclic property), we deduce that $\xi\Sigma\Sigma'\Phi = \xi\Sigma\Sigma\Sigma'\Phi$ and $u\sigma\sigma' = u\sigma\sigma\sigma'$. Hence, $Msg(\xi\Sigma\Sigma'\Phi)$ is equivalent to $Msg(\xi\Sigma\Sigma\Sigma'\Phi)$; and $\xi\Sigma\Sigma'\Phi{\downarrow} = u\sigma\sigma'$ is equivalent to $\xi\Sigma\Sigma\Sigma'\Phi{\downarrow} = u\sigma\sigma\sigma'$. Hence $(\Phi, \Sigma\Sigma', \sigma\sigma') \models \xi \vdash^? u$ is equivalent to $(\Phi, \Sigma\Sigma', \sigma\sigma') \models \xi \vdash^? u\Sigma\sigma$. Note that $Msg(\xi\Sigma\Sigma'\Phi)$ and $\xi\Sigma\Sigma'\Phi{\downarrow} = u\sigma\sigma'$ are also equivalent to $(\Phi, \Sigma', \sigma') \models \xi\Sigma \vdash^? u\sigma$.

Case $\xi =^?_f \xi'$: Similar to previous case. □

B.3 Properties on consequence of set of deduction facts

Lemma 23. *Let S be a set of solved deduction facts. For all substitutions σ of protocol terms, for all $(\xi, t) \in$ Conseq(S), $(\xi, t\sigma) \in$ Conseq$(\mathsf{S}\sigma)$.*

Proof. We know that $(\xi, t) \in \textsc{Conseq}(\mathsf{S})$ implies $\xi = C[\xi_1, \ldots, \xi_n]$ and $t = C[t_1, \ldots, t_n]$ for some public context C and for all i, $(\xi_i \vdash^? t_i \in \mathsf{S}$. Hence $(\xi_i \vdash^? t_i\sigma \in \mathsf{S}\sigma$ which allows us to conclude. □

Lemma 14. *Let* S, S' *be two sets of solved deduction facts. Let* $\varphi = \{X_i \vdash^? u_i\}_{i=1}^n$ *such that all* X_i *are pairwise distinct. For all* Σ, σ, *for all* $(\xi, t) \in \textsc{Conseq}(\mathsf{S} \cup \varphi)$, *if for all* $i \in \{1, \ldots, n\}$, $(X_i\Sigma, u_i\sigma) \in \textsc{Conseq}(\mathsf{S}\Sigma\sigma \cup \mathsf{S}')$ *then* $(\xi\Sigma, t\sigma) \in \textsc{Conseq}(\mathsf{S}\Sigma\sigma \cup \mathsf{S}')$.

Proof. We prove this result by induction on $|\xi|$. The base case ($|\xi| = 0$) being trivial as there is no term of size 0, we focus on the inductive step.

Since (ξ, t) is consequence of $\mathsf{S} \cup \varphi$, we know by definition that one of the following conditions hold:

1. $\xi = t \in \mathcal{N}_{\mathsf{pub}}$
2. there exists $\xi_1, t_1, \ldots, t_m, \xi_m$ and $\mathsf{f} \in \mathcal{F}_c$ such that $\xi = \mathsf{f}(\xi_1, \ldots, \xi_m)$, $t = \mathsf{f}(t_1, \ldots, t_m)$ and for all $i \in \{1, \ldots, m\}$, (ξ_i, t_i) is consequence of $\mathsf{S} \cup \varphi$.
3. $\xi \vdash^? t \in \mathsf{S} \cup \varphi$.

In case 1, we directly have by definition that $(\xi\Sigma, t\sigma)$ is a consequence of $\mathsf{S}\Sigma\sigma \cup \mathsf{S}'$. In case 2, by our inductive hypothesis on $\xi_1, \ldots, \xi_m$, we have that for all $j \in \{1, \ldots, m\}$, $(\xi_j\Sigma, t_j\sigma)$ is a consequence of $\mathsf{S}\Sigma\sigma \cup \mathsf{S}'$. With $\xi\Sigma = \mathsf{f}(\xi_1\Sigma, \ldots, \xi_m\Sigma)$ and $t\sigma = \mathsf{f}(t_1\sigma, \ldots, t_m\sigma)$, we conclude that $(\xi\Sigma, t\sigma)$ is consequence of $\mathsf{S}\Sigma\sigma \cup \mathsf{S}'$. In case 3, if $\xi \vdash^? t \in \mathsf{S}$ then $\xi\Sigma \vdash^? t\sigma \in \mathsf{S}\Sigma\sigma$ and so the result directly holds. Else $\xi \vdash^? t \in \varphi$ and so by hypothesis $\xi\Sigma \vdash^? t\sigma \in \textsc{Conseq}(\mathsf{S}\Sigma\sigma \cup \mathsf{S}')$. □

The previous lemma showed that a consequence (ξ, t) is preserved when applying some substitution Σ, σ under the right conditions. However, it is quite strong since we ensure that $\xi\Sigma$ is consequence with $t\sigma$. In some cases, we cannot guarantee that $\xi\Sigma$ is consequence with $t\sigma$ but with some other first-order term. This is the purpose of the next lemma.

Lemma 24. *Let* S, S' *be two sets of solved deduction facts. Let* $\varphi = \{X_i \vdash^? u_i\}_{i=1}^n$ *such that all* X_i *are pairwise distinct. For all* Σ, *for all* $\xi \in \textsc{Conseq}(\mathsf{S} \cup \varphi)$, *if for all* $i \in \{1, \ldots, n\}$, $X_i\Sigma \in \textsc{Conseq}(\mathsf{S}\Sigma \cup \mathsf{S}')$ *then* $\xi\Sigma \in \textsc{Conseq}(\mathsf{S}\Sigma \cup \mathsf{S}')$.

Proof. We prove this result by induction on $|\xi|$. The base case ($|\xi| = 0$) being trivial as there is no term of size 0, we focus on the inductive step.

Since ξ is consequence of $\mathsf{S} \cup \varphi$, we know by definition that there exists t such that one of the following conditions hold:

1. $\xi = t \in \mathcal{N}_{\mathsf{pub}}$
2. there exists $\xi_1, t_1, \ldots, t_m, \xi_m$ and $\mathsf{f} \in \mathcal{F}_c$ such that $\xi = \mathsf{f}(\xi_1, \ldots, \xi_m)$, $t = \mathsf{f}(t_1, \ldots, t_m)$ and for all $i \in \{1, \ldots, m\}$, (ξ_i, t_i) is consequence of $\mathsf{S} \cup \varphi$.
3. there exists t such that $\xi \vdash^? t \in \mathsf{S} \cup \varphi$.

In case 1, we directly have by definition that $(\xi\Sigma, t)$ is a consequence of $\mathsf{S}\Sigma \cup \mathsf{S}'$. In case 2, by our inductive hypothesis on $\xi_1, \ldots, \xi_m$, we have that for all $j \in \{1, \ldots, m\}$, $\xi_j\Sigma$ is a consequence of $\mathsf{S}\Sigma \cup \mathsf{S}'$ hence there exists $t'_1, \ldots, t'_m$ such that for all $j \in \{1, \ldots, m\}$, $(\xi_j\Sigma, t'_j)$ is a consequence of $\mathsf{S}\Sigma \cup \mathsf{S}'$. With $\xi\Sigma = \mathsf{f}(\xi_1\Sigma, \ldots, \xi_m\Sigma)$ and $t' = \mathsf{f}(t'_1, \ldots, t'_m)$, we conclude that $(\xi\Sigma, t')$ is consequence of $\mathsf{S}\Sigma \cup \mathsf{S}'$. In case 3, if $\xi \vdash^? t \in \mathsf{S}$ then $\xi\Sigma \vdash^? t \in \mathsf{S}\Sigma$ and so the result directly holds. Else $\xi \vdash^? t \in \varphi$ and so by hypothesis $\xi\Sigma \in \textsc{Conseq}(\mathsf{S}\Sigma \cup \mathsf{S}')$. □

In the next lemma, we show that when a recipe is consequence of the sets of solved deduction formulas $\mathsf{K}\Sigma\sigma$ where (Σ, σ) is a solution of the constraint system, then all subterms of that recipe are also consequence of $\mathsf{K}\Sigma\sigma$. This property is in fact guaranted by the fact that K contains itself recipes consequence of itself. This is an important property that allows us to generate solutions that satisfy the uniformity property.

Lemma 25. *Let $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ be an extended constraint system such that $\mathcal{P}_{form}(\mathcal{C})$. For all $\xi \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$, for all $\xi' \in st(\xi)$, $\xi' \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$.*

Proof. Since $\xi \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$, we know that $\xi = C[\xi_1, \ldots, \xi_n]$ where C is a public context and $\xi_1, \ldots, \xi_n$ are recipes of deduction facts from K or D. Hence since $\xi' \in st(\xi)$, we have that the position p of ξ' in ξ is either a position of C thus $\xi' \in\in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$ from the definition of consequence; or is a position of one of the ξ_i and thus we conclude by the predicate $\mathcal{P}_{form}(\mathcal{C})$. □

Lemma 26. *Let $\mathcal{C} = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$ be an extended constraint system such that $\mathcal{P}_{form}(\mathcal{C})$. For all $(\Sigma, \sigma) \in Sol_e(\mathcal{C})$, for all $\xi \in \textsc{Conseq}(\mathsf{K}\Sigma\sigma)$, for all $\xi' \in st(\xi)$, $\xi' \in \textsc{Conseq}(\mathsf{K}\Sigma\sigma)$.*

Lemma 27. *Let S be a set of ground deduction facts. Let Φ be a ground frame. Assume that for all $\psi \in \mathsf{S}$, $\Phi \models \psi$. For all $(\xi, t) \in \textsc{Conseq}(\mathsf{S})$, $\Phi \models \xi \vdash^? t$.*

Proof. We prove this result by induction on $|\xi|$. The base case being trivial, we focus on the inductive step. Since (ξ, t) is consequence of S then one of the following properties holds:

1. $\xi = t \in \mathcal{N}_{\mathsf{pub}}$
2. there exist $\xi_1, t_1, \ldots, \xi_n, t_n$ and $\mathsf{f} \in \mathcal{F}_c$ such that $\xi = \mathsf{f}(\xi_1, \ldots, \xi_n)$, $t = \mathsf{f}(t_1, \ldots, t_n)$ and for all $i \in \{1, \ldots, n\}$, (ξ_i, t_i) is consequence of S
3. $\xi \vdash^? t \in \mathsf{S}$.

In Case 1, the result trivially holds. In case two, a simple induction on $(\xi_1, t_1), \ldots, (\xi_n, t_n)$ allows us to conclude. In case 3, we know by hypothesis that $\Phi \models \xi \vdash^? t$ hence the result holds □

B.4 Most general solutions

In this section, we show that given an extended constraint system $\mathcal{C}$, the rules 8 to 12 and appendix A.1.2 allows to compute the the most general solutions of $\mathcal{C}$.

Given a constraint system $\mathcal{C}$, let us denote $R(\mathcal{C})$ the set $st^2(img(mgu(\mathsf{E}^2(\mathcal{C})))) \cup sst^2(\mathsf{K}(\mathcal{C})) \cup vars^2(\mathsf{D}(\mathcal{C}))$

Lemma 28. *Let $\mathcal{C}$ an extended constraint system such that $\mathcal{P}_{form}(\mathcal{C})$ and $\mathcal{P}_{sound}(\mathcal{C})$. If any rule is applicable on $\mathcal{C}$ then for all $(\Sigma, \sigma) \in Sol_e(\mathcal{C})$, there exists $\mathcal{C}'$, Σ' such that $\mathcal{C} \Rightarrow\!\wr\, \mathcal{C}'$ and $(\Sigma\Sigma', \sigma) \in Sol_e(\mathcal{C}')$.*

Proof. First, assume that there exist $\xi, \zeta \in R(\mathcal{C})^2$ and u such that $\xi \neq \zeta$ and $(\xi, u), (\zeta, u) \in$ CONSEQ$(\mathsf{K} \cup \mathsf{D})$. Thanks to $\mathcal{P}_{sound}(\mathcal{C})$ and lemmas 14 and 27, we deduce that $\Phi\sigma \models \xi\Sigma \vdash^? u\sigma \wedge \zeta\Sigma \vdash^? u\sigma$. As such we have $\xi\Sigma{\downarrow} = u\sigma = \zeta\Sigma{\downarrow}$. However by definition of a solution it implies that $\xi\Sigma = \zeta\Sigma$. Thus there exists $\Sigma' = mgu(\xi =^? \zeta)$ such that $\Sigma' \neq \emptyset$ and $\Sigma' \neq \bot$.

In such a case, let us show that $(\Sigma, \sigma) \in Sol_e(\mathcal{C}')$ with $\mathcal{C} \to \mathcal{C}'$ by rule 8. We already know that $\Sigma \models \mathsf{E}^2(\mathcal{C})$ and since $\xi\Sigma = \zeta\Sigma$ with $\Sigma' = mgu(\xi =^? \zeta)$, we directly have that $\Sigma \models \mathsf{E}^2(\mathcal{C})\Sigma' \wedge \Sigma'$. Moreover, $\mathsf{K}(\mathcal{C})\Sigma = \mathsf{K}(\mathcal{C})\Sigma'\Sigma$. Hence, the two bullets of the definition of solutions is trivially satisfied by that fact that $(\Sigma, \sigma) \in Sol_e(\mathcal{C})$. Therefore, we conclude that $(\Sigma, \sigma) \in Sol_e(\mathcal{C}')$.

Let us now consider the case where our assumption do no hold. Thus since we assume that at least one rule is applicable on $\mathcal{C}$, there exists $X \vdash^? u \in \mathsf{D}(\mathcal{C})$ where $u \notin \mathcal{X}^1$. Let us do a case analysis on $X\Sigma$ since $X\Sigma \in$ CONSEQ$(\mathsf{K}\Sigma\sigma)$ by definition of a solution.

- either $X\Sigma \in \mathcal{N}_{\mathsf{pub}}$: in such a case, we have $\mathcal{C} \to \mathcal{C}'$ by rule 8 and we can prove similarly as in the previous case that $(\Sigma, \sigma) \in Sol_e(\mathcal{C}')$;
- or $X\Sigma = \mathsf{f}(\xi_1, \ldots, \xi_n)$ where $\xi_i \in$ CONSEQ$(\mathsf{K}\Sigma\sigma)$ for all i: Note that we know that $X\Sigma\Phi\sigma{\downarrow} = u\sigma$. Hence $u = \mathsf{f}(u_1, \ldots, u_n)$ for some $u_1, \ldots, u_n$. We deduce that for all i, $\Phi\sigma \models \xi_i \vdash^? u_i\sigma$. Thus, by considering $\Sigma' = \{X_i \mapsto \xi_i\}_{i=1}^n$, we can conclude that $\mathcal{C} \to \mathcal{C}'$ by rule 9 and $(\Sigma\Sigma', \sigma) \in Sol_e(\mathcal{C}')$;
- or $X\Sigma \vdash^? u\sigma \in \mathsf{K}\Sigma\sigma$ (since once again $X\Sigma\Phi\sigma{\downarrow} = u\sigma$): Thus there exists $\xi \vdash^? v \in \mathsf{K}$ such that $\xi\Sigma = X\Sigma$ and $u\sigma = v\sigma$. Hence $mgu(\xi, X)$ exists and $\sigma \models u =^? v$. Thereofore, we can conclude that $\mathcal{C} \to \mathcal{C}'$ by rule 10 and $(\Sigma, \sigma) \in Sol_e(\mathcal{C}')$;

□

Lemma 29. *Let $\mathcal{C} \neq \bot$ an extended constraint system such that $\mathcal{C}\!\wr = \mathcal{C}$, $\mathcal{P}_{form}(\mathcal{C})$ and $\mathcal{P}_{sound}(\mathcal{C})$. If $\mathcal{C} \not\Rightarrow\!\wr$ and $\mathcal{C}$ is a solved extended constraint system then $mgs(\mathcal{C}) = \{mgu(\mathsf{E}^2|_=)\}$.*

Proof. We know that for all $(\Sigma, \sigma) \in Sol_e(\mathcal{C})$, $\Sigma \models \mathsf{E}^2(\mathcal{C})$ thus we directly obtain the existence of Σ' such that $\Sigma = mgu(\mathsf{E}^2|_=)\Sigma'$. Consider now the second bullet point of the definition of most general solutions. We know that $\mathcal{C}$ is solved. Hence consider a fresh bijective renaming Σ_1 from $vars^2(\Sigma_0) \cup vars^2(\mathcal{C}) \setminus dom(\Sigma_0)$ to $\mathcal{N}_{\mathsf{pub}}$. Let us define $\sigma_1 = \{x \mapsto X\Sigma_1 \mid X \vdash^? x\mathsf{D}(\mathcal{C})\}$. Thanks to $\mathcal{P}_{form}(\mathcal{C})$, $\mathcal{P}_{sound}(\mathcal{C})$ and lemmas 24, 26 and 27 that $(\Phi mgu(\mathsf{E}^1(\mathcal{C})|_=)\sigma_1, mgu(\mathsf{E}^2|_=)\Sigma_1, mgu(\mathsf{E}^1(\mathcal{C})|_=)\sigma_1) \models \mathsf{D} \wedge \mathsf{E}^1 \wedge \mathsf{E}^2$. Moreover, by lemma 26, we know that the first bullet of the definition of solution is satisfied. Finally, the second bullet is satisfied otherwise rule 12 would be applicable which contradict $\mathcal{C}\!\wr = \mathcal{C}$. Therefore, $(mgu(\mathsf{E}^2|_=)\Sigma_1, mgu(\mathsf{E}^1(\mathcal{C})|_=)\sigma_1) \in Sol_e(\mathcal{C})$. We conclude that $mgs(\mathcal{C}) = \{mgu(\mathsf{E}^2|_=)\}$. □

Lemma 30. *Let $\mathcal{C}$ an extended constraint system such that $\mathcal{C}\!\wr = \mathcal{C}$, $\mathcal{P}_{form}(\mathcal{C})$ and $\mathcal{P}_{sound}(\mathcal{C})$. If $\mathcal{C} \not\Rightarrow\!\wr$ and $\mathcal{C}$ is not solved then $Sol_e(\mathcal{C}) = \emptyset$.*

Proof. Since $\mathcal{C}$ is not solved, we have two possibilities: Either (a) all deduction facts in D are have variables as right hand term but not pairwise distinct. But in such a case rule 8 would be applicable which contradicts $\mathcal{C} \not\rightarrow_{\wr}$; or (b) there exists $(X \vdash^? u) \in \mathsf{D}(\mathcal{C})$ such that $u \not\in \mathcal{X}^1$. Since rule 8 is not applicable, we deduce that $u \not\in \mathcal{N}_{\mathsf{pub}}$ and for all $\xi, \zeta \in \mathsf{R}(\mathcal{C}) \setminus \{X\}$, $(\xi, u) \not\in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$. But rule rule 9 is not applicable therefore, we deduce that $u \in \mathcal{N}_{\mathsf{prv}}$.

Assume now that $Sol_e(\mathcal{C}) \neq \emptyset$ and so $(\Sigma, \sigma) \in Sol_e(\mathcal{C})$. Thus $X\Sigma\Phi{\downarrow} = u$. By definition of a solution, we know that $(X\Sigma, u) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C})\Sigma\sigma)$. Since $u \in \mathcal{N}_{\mathsf{prv}}$ it implies that there exists $(\xi \vdash^? v) \in \mathsf{K}(\mathcal{C})$ such that $X\Sigma = \xi\Sigma$ and $u = v\sigma$. Note that by $\mathcal{P}_{form}(\mathcal{C})$, we also have that $v \not\in \mathcal{X}^1$ and so $u = v$. In such a case, we obtain a contradiction with the fact the rule 10 is not applicable. □

By combining the previous three lemmas, we can directly derive the following lemma:

Lemma 13. *Let $\mathcal{C}$ be an extended constraint system obtained during the procedure*

$$mgs(\mathcal{C}) = \{\Sigma|_{vars^2(\mathcal{C})} \mid \mathcal{C} \overset{\Sigma}{\Rightarrow}_{\wr} \mathcal{C}', \mathcal{C}' \neq \bot, \mathcal{C}' \text{ solved}\}$$

B.5 Properties of the partition tree

In this section we focus on the different properties of that must be satisfied by the partition tree.

Let us first start by noticing that the case distinction rules and simplification rules preserves the first order solutions of the extended constraint systems. This property is stated in the following lemma.

Lemma 31. *Let $\mathcal{S}$ be a set of set of extended symbolic processes such that $\mathcal{P}_{all}(\mathcal{S})$. Let $\mathcal{S} \rightarrow \mathcal{S}'$ by applying only case distinction or simplifications rules (i.e. no symbolic transitions) following the priority order* Sat < Equality < Rew. *We have that $\mathcal{P}_{all}(\mathcal{S}')$ and:*

- *for all $S \in \mathcal{S}$, for all $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e) \in S$, for all $(\Sigma, \sigma) \in Sol_e(\mathcal{C}^e)$, there exist $S' \in \mathcal{S}'$, $(\mathcal{P}, \mathcal{C}, \mathcal{C}^{e\prime}) \in S'$ and $(\Sigma', \sigma') \in Sol_e(\mathcal{C}^{e\prime})$ such that $\sigma|_{vars^1(\mathcal{C})} = \sigma'|_{vars^1(\mathcal{C})}$*
- *for all $S' \in \mathcal{S}$, for all $(\mathcal{P}, \mathcal{C}, \mathcal{C}^{e\prime}) \in S'$, for all $(\Sigma', \sigma') \in Sol_e(\mathcal{C}^{e\prime})$, there exist $S \in \mathcal{S}$, $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e) \in S$ and $(\Sigma, \sigma) \in Sol_e(\mathcal{C}^e)$ such that $\Sigma|_{vars^2(\mathcal{C})} = \Sigma'|_{vars^2(\mathcal{C})}$ and $\sigma|_{vars^1(\mathcal{C})} = \sigma'|_{vars^1(\mathcal{C})}$*

Proof. We do a case analysis on the rule applied. Let us look first at the simplification rules.

First, let us notice the result directly hold for rules 13, 16 and 17. Indeed, rule 15 does not modify constraints on recipe and preserves the constraints on protocol terms. Moreover, rules 16 and 17 affect F which do not impact the solutions of the extended constraint system. For rule 14, since $mgs(\mathcal{C}^e) = \emptyset$, we have by definition of most general unifiers that $Sol_e(\mathcal{C}^e) = \emptyset$ (otherwise the first bullet of the definition is contradicted). Hence the result holds since $Sol_e(\bot) = \emptyset$. Similarly, Rule 16 checks whether the disequations $\forall\tilde{x}.\phi$ is trivially true meaning that the rule preserves the solutions.

Let us now consider the simplification rules on sets of extended symbolic processes. Rule 18 only remove an extended symbolic process with an extended constraint systems having no solution hence the result hold. Rule 19 splits a set of $\mathcal{S}$ into two sets thus preserving the extended symbolic processes, and rule 21 only adds element in F which do not impact the solutions of a constraint system. Therefore, for all these rules, the result hold. For rule 20 however, the result is not direct since the rule adds an element in the set K which has an impact on the solutions of a constraint system. However, we know from the application condition of the rule that the head protocol terms of the deduction facts added in K_i are not consequence of $\mathsf{K}_i \cup \mathsf{D}_i$. But we also know that $\mathcal{P}_{cons}(\mathcal{S})$ and $\mathcal{P}_{form}(\mathcal{S})$ hold hence it implies that the recipe ξ (see fig. 14) contains $\mathsf{ax}_{|\Phi_i|}$ and $vars^2(\mathsf{D}_i) \cap \mathcal{X}^2_{|\Phi_i|} = \emptyset$. Hence, ξ cannot appear in the second order solutions $\mathcal{C}_i^e$ which allows us to conclude that the solutions are preserved.

Finally, the case distinction rules apply a perfect partition of the solutions by duplicating the set of extended symbolic processes and applying Σ on of the set and $\neg\Sigma$ on the other (see the definition of rule Sat, Equality and Rew). □

We can also show that the static equivalence is preserved by application of the case distinction and simplification rules.

Lemma 32. *Let $\mathcal{S}$ be a set of set of extended symbolic processes such that $\mathcal{P}_{all}(\mathcal{S})$. Let $\mathcal{S} \to \mathcal{S}'$ by applying only case distinction or simplifications rules (i.e. no symbolic transitions) following the priority order* Sat $<$ Equality $<$ Rew. *We have that for all $S \in \mathcal{S}$, for all $(\mathcal{P}_1, \mathcal{C}_1, \mathcal{C}_1^e), (\mathcal{P}_2, \mathcal{C}_2, \mathcal{C}_2^e) \in S$, for all $(\Sigma, \sigma_1) \in Sol_e(\mathcal{C}_1^e)$, for all $(\Sigma, \sigma_2) \in Sol_e(\mathcal{C}_2^e)$, if $\Phi(\mathcal{C}_1)\sigma_1 \sim \Phi(\mathcal{C}_2)\sigma_2$ then for all $S' \in \mathcal{S}'$, for all Σ' such that $\Sigma'|_{vars^2(\mathcal{C}_1)} = \Sigma|_{vars^2(\mathcal{C}_1)}$,*

$$(\mathcal{P}_1, \mathcal{C}_1, \mathcal{C}_1^{e\prime}) \in S' \wedge (\Sigma', \sigma_1') \in Sol_e(\mathcal{C}_1^{e\prime}) \wedge \sigma_1|_{vars^1(\mathcal{C}_1)} = \sigma_1'|_{vars^1(\mathcal{C}_1)}$$
$$\textit{if and only if}$$
$$(\mathcal{P}_2, \mathcal{C}_2, \mathcal{C}_2^{e\prime}) \in S' \wedge (\Sigma', \sigma_2') \in Sol_e(\mathcal{C}_2^{e\prime}) \wedge \sigma_2|_{vars^1(\mathcal{C}_2)} = \sigma_2'|_{vars^1(\mathcal{C}_2)}$$

Proof. Once again, let us consider the potential rule applied. Note that for case distinction rules, the proof is simple since each rule create a partition of the second-order solutions with respect to some substitution Σ_0 corresponding to a most general solutions (see the definition of rule Sat, Equality and Rew). Thus, assume w.l.o.g. that $(\Sigma', \sigma_1') \in Sol_e(\mathcal{C}_1^{e\prime})$.

First consider that S' corresponds to branch in which we applied $\neg\Sigma_0$. In such a case, since we already know that Σ' satisfies $\neg\Sigma_0$ and no other constraint is added, we directly obtain from $(\Sigma, \sigma_2) \in Sol_e(\mathcal{C}_2^e)$ that $(\Sigma', \sigma_2') \in Sol_e(\mathcal{C}_2^{e\prime})$ (in this case, we even have $\sigma_2 = \sigma_2'$.

Now consider that S' corresponds to the branch in which we applied Σ_0. In such a case, the application of Σ_0 on $\mathcal{C}_2^e$ regroups all the solution of $\mathcal{C}_2^e$ that satisfies Σ_0. Since we know that $(\Sigma, \sigma_2) \in Sol_e(\mathcal{C}_2^e)$ and $\Sigma'|_{vars^2(\mathcal{C}_1)} = \Sigma|_{vars^2(\mathcal{C}_1)}$ which implies $\Sigma'|_{vars^2(\mathcal{C}_2)} = \Sigma|_{vars^2(\mathcal{C}_2)}$, the result holds.

Concerning the simplification rules, the only problematic one is rule 19 (the other ones do not split the set). However, thanks to $\mathcal{P}_{vect}(\mathcal{S})$, we know that if a deduction fact occurs in constraint systems $\mathcal{C}_1^e$ but no recipe equivalent formula can be found in the constraint system $\mathcal{C}_2^e$, then no solution of $\mathcal{C}_2^e$ can satisfy the head of the formula. However by $\mathcal{P}_{comp}(\mathcal{S})$, we also know that all solutions of $\mathcal{C}_1^e$ satisfies the deduction fact. However, since $(\Sigma, \sigma_1) \in Sol_e(\mathcal{C}_1^e)$, $(\Sigma, \sigma_2) \in Sol_e(\mathcal{C}_2^e)$ and $\Phi(\mathcal{C}_1)\sigma_1 \sim \Phi(\mathcal{C}_2)\sigma_2$, we obtain a contradiction. Therefore, $\mathcal{C}_1^e$ and $\mathcal{C}_2^e$ are necessarily in the set of $\mathcal{S}'$. □

The two previous lemmas allow us to obtain the soundness and completeness properties of the partition tree (i.e. items 3 and 5). Note that 4 is also proved by 31 since a solution of a child constraint system is also a solution of parent one (second bullet point). We now need to prove that all nodes of the partition tree are configuration. For that, we look at the shape of the extended constraint systems when no more rule SAT, EQUALITY or REW is applicable.

Lemma 33. *Let $\mathcal{S}$ be a set of set of extended symbolic processes such that $\mathcal{P}_{all}(\mathcal{S})$ and no instance of the rule* SAT, EQUALITY *or* REW *is applicable. For all $S \in \mathcal{S}$, for all $(\mathcal{P}, \mathcal{C}, \mathcal{C}^e) \in S$, $|mgs(\mathcal{C}^e)| = 1$.*

Proof. Since no rule is applicable, we need to look at the application conditions of each of these rules. Let us denote $C^e = (\Phi, \mathsf{D}, \mathsf{E}^1, \mathsf{E}^2, \mathsf{K}, \mathsf{F})$.

Let us look at the shape of F. Since the rule SAT is not applicable (case b), we know that for all $\psi \in \mathsf{F}$, either ψ is solved or $mgs(\mathcal{C}^e|^{\psi}) = \emptyset$. However, since the normalisation rules from fig. 13 are also not applicable, we deduce that $mgs(\mathcal{C}^e|^{\psi}) \neq \emptyset$. Therefore, ψ is solved. Similarly, the non applicability of the rule SAT (case c) and the normalisation rules from fig. 13 also tell us that E^1 is only composed of syntactic equations (no more disequations). Finally, since SAT is not applicable (case a), we deduce that either $\mathcal{C}^e$ is solved or $mgs(\mathcal{C}^e) = \emptyset$. Once again due to rules fig. 13, we deduce that $mgs(\mathcal{C}^e) \neq \emptyset$ meaning that $\mathcal{C}^e$ is solved.

Since $\mathcal{C}^e$ is solved, we deduce that all deduction facts in $\mathsf{D} = \{X_i \vdash^? x_i\}_{i=1}^n$ for some n and pairwise distinct x_is and X_is. Consider now the substitutions $\Sigma_{\mathcal{N}_{\mathsf{pub}}} = \{X_i \to n_i\}_{i=1}^n$ and $\sigma_{\mathcal{N}_{\mathsf{pub}}} = \{x_i \to n_i\}_{i=1}^n$ where the n_is are pairwise distincts public names, i.e. $n_i \in \mathcal{N}_{\mathsf{pub}}$. Since no more normalisation rules are applicable, we know that the disequations in E^2 not trivially unsatisfiable. Therefore by replacing the free variables of the disequations by names allow us to obtain that $\Sigma_{\mathcal{N}_{\mathsf{pub}}} \models \mathsf{E}^2|_{\neq}$. By considering $\Sigma = mgu(\mathsf{E}^2|_{=})\Sigma'$, we obtain that $\Sigma \models \mathsf{E}^2$. Moreover we proved that E^1 does not contain any disequations, we directly obtain that $mgu(\mathsf{E}^1)\sigma_{\mathcal{N}_{\mathsf{pub}}} \models \mathsf{E}^1$. Therefore, by defining $\sigma = mgu(\mathsf{E}^1)\sigma_{\mathcal{N}_{\mathsf{pub}}}$, we obtain that $(\Phi\sigma, \Sigma, \sigma) \models \mathsf{D} \wedge \mathsf{E}^1 \wedge \mathsf{E}^2$.

It remains to show the last two conditions of definition 23: Notice that $\mathcal{P}_{form}(\mathcal{C}^e)$ holds and so we have that for all $\xi \in img(mgu(\mathsf{E}^2|_{=}))$, $\xi \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$. Similarly, we have that for all $\xi \vdash^? u \in \mathsf{K}$, for all $\xi' \in st(\xi)$, $\xi' \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$. Therefore, by applying lemma 24, and by applying a small induction on the size of the recipe in $img(\Sigma)$, we obtain that for all $\xi \in st(img(\Sigma))$, $\xi \in \textsc{Conseq}(\mathsf{K}\Sigma)$. Finally, following the definition of consequence and since $\mathcal{P}_{sound}(\mathcal{C}^e)$ holds, we also deduce that $\xi \in st(img(\Sigma))$, $Msg(\xi\Phi\sigma)$ holds.

Let us now look at the final condition of definition 23. We know that $\mathcal{C}^e$ is solved meaning that for all $\xi, \zeta \in st_c(img(mgu(\mathsf{E}^2|_{=})), \mathsf{K} \cup \mathsf{D})^2 \cup (\mathcal{N}_{\mathsf{pub}} \times vars^2(\mathsf{D}))$, $(\xi, u), (\zeta, u) \in \textsc{Conseq}(\mathsf{K} \cup \mathsf{D})$ implies $\xi = \zeta$. Since $\Sigma = mgu(\mathsf{E}^2|_{=})\Sigma_{\mathcal{N}_{\mathsf{pub}}}$, we directly obtain that for all $\xi, \zeta \in st_c(\Sigma, \mathsf{K}\Sigma)$, $(\xi, u), (\zeta, u) \in \textsc{Conseq}(\mathsf{K}\Sigma)$ implies $\xi = \zeta$.

This lead to $(\Sigma, \sigma) \in Sol_e(\mathcal{C}^e)$. Considering that $\Sigma = mgu(\mathsf{E}^2|_{=})\Sigma_{\mathcal{N}_{\mathsf{pub}}}$ and any other solutions $(\Sigma', \sigma') \in Sol_e(\mathcal{C}^e)$ shall satisfy $\Sigma' \models \mathsf{E}^2$, we deduce that $mgs(\mathcal{C}^e) = \{mgu(\mathsf{E}^2|_{=})\}$ and so $|mgs(\mathcal{C}^e)| = 1$. □

Let us now show that all extended constraint systems in the set have the same solutions and that they are statically equivalent.

Lemma 34. *Let $\mathcal{S}$ be a set of set of extended symbolic processes such that $\mathcal{P}_{all}(\mathcal{S})$ and no instance of the rule* SAT, EQUALITY *or* REW *is applicable. For all $S \in \mathcal{S}$, for all*

$(\mathcal{P}_1, \mathcal{C}_1, \mathcal{C}_1^e), (\mathcal{P}_2, \mathcal{C}_2, \mathcal{C}_2^e) \in S$, *if* $(\Sigma, \sigma_1) \in \mathit{Sol}_e(\mathcal{C}_1^e)$ *then* $(\Sigma, \sigma_2) \in \mathit{Sol}_e(\mathcal{C}_2^e)$ *and* $\Phi(\mathcal{C}_1^e)\sigma_1 \sim \Phi(\mathcal{C}_2^e)\sigma_2$.

Proof. In the proof of the previous lemma, we showed that all extended constraint systems $\mathcal{C}^e$ in S have a particular form, that is (1) all deduction facts in $\mathsf{D}(\mathcal{C}^e)$ have pairwise distinct variables as right hand side and (2) $\mathsf{E}^1(\mathcal{C}^e)$ only contain syntactic equation. Moreover, we know that all extended constraint systems have the same structure. Therefore, if $(\Sigma, \sigma_1) \in \mathit{Sol}_e(\mathcal{C}_1^e)$, we deduce that $\Sigma \models \mathsf{E}^2(\mathcal{C}_1^e)$ and for all $\xi \in \mathit{st}(\mathit{img}(\Sigma))$, $\xi \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$, meaning that $\Sigma \models \mathsf{E}^2(\mathcal{C}_2^e)$ and $\xi \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_2^e)\Sigma)$. Since the first order solutions are always completely defined by the second-order substitutions, we can build σ_2' such that for all $X \vdash^? x \in \mathsf{D}(\mathcal{C}_2^e)$, $X\Sigma(\Phi(\mathcal{C}_2^e)\sigma_2')\!\downarrow = x\sigma_2'$. Moreover, since $\mathcal{P}_{sound}(\mathcal{C}_1^e)$ and $\mathcal{P}_{sound}(\mathcal{C}_2^e)$ both hold and since for all $\xi \in \mathit{st}(\mathit{img}(\Sigma))$, $\xi \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_2^e)\Sigma)$, we deduce that for all $\xi \in \mathit{st}(\mathit{img}(\Sigma))$, $\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2')$. Note that we also need to satisfy the syntactic equations in E^1. However thanks to $\mathcal{P}_{form}(\mathcal{C}_2^e)$ holding, we know that $\mathit{dom}(\mathit{mgu}(\mathsf{E}^1(\mathcal{C}_2^e)) \cap \mathit{vars}^1(\mathsf{D}(\mathcal{C}_2^e)) = \emptyset$. Thus, we can build $\sigma_2 = \mathit{mgu}(\mathsf{E}^1(\mathcal{C}_2^e))\sigma_2'$ and obtain that $(\Sigma, \sigma_2) \models \mathsf{D}(\mathcal{C}_2^e) \wedge \mathsf{E}^1(\mathcal{C}_2^e) \wedge \mathsf{E}^2(\mathcal{C}_2^e)$. Note that by origination property of an extended constraint system, we have $\Phi(\mathcal{C}_2^e)\sigma_2' = \Phi(\mathcal{C}_2^e)\sigma_2$. Therefore, since we already prove that for all $\xi \in \mathit{st}(\mathit{img}(\Sigma))$, $\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2')$ and $\xi \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_2^e)\Sigma)$, it only remains to prove the second bullet point of definition 23 to obtain that $(\Sigma, \sigma_2) \in \mathit{Sol}_e(\mathcal{C}_2^e)$.

Proving this property is in fact closely related to proving that $\Phi(\mathcal{C}_1^e)\sigma_1 \sim \Phi(\mathcal{C}_2^e)\sigma_2$. Indeed, if we can prove the latter then we directly obtain the former since $(\Sigma, \sigma_1) \in \mathit{Sol}_e(\mathcal{C}_1^e)$.

Let $\xi, \xi' \in \mathcal{T}^2$. Let us show that:

- $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$ iff $\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2)$
- if $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$ and $\mathit{Msg}(\xi'\Phi(\mathcal{C}_1^e)\sigma_1)$ then $\xi\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow = \xi'\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$ iff $\xi\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow = \xi'\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow$.

We prove this by induction on $(N(\xi, \xi'), |\xi\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow|)$ where N is the number of subterms ζ in ξ, ξ' such that $\zeta \notin \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$ (recall that since $\mathcal{C}_1^e$ and $\mathcal{C}_2^e$ have the same structure, we have $\zeta \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$ iff $\zeta \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_2^e)\Sigma)$). For this induction, we consider the lexicographic order on $(N(\xi, \xi'), \max(|\xi|, |\xi'|))$.

The base case $(N(\xi, \xi'), |\xi\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow|) = (0, 0)$ being trivial (there is no protocol term of size 0), we focus on the two inductive steps:

Inductive steps $(N(\xi, \xi'), |\xi\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow|) = (p, q)$ with $p > 0$: Assume $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$. Let us also assume by contradiction that $\neg\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2)$. Since we know that $N(\xi, \xi') > 0$, there exists $\zeta \in \mathit{st}(\xi, \xi')$ such that $\zeta \notin \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)$. Without lost of generality we can consider that $\zeta \in \mathit{st}(\xi)$ (otherwise we can apply our inductive hypothesis on ξ twice since $N(\xi, \xi)$ would be equal to 0 and so we would obtain a contradiction). Moreover, let us consider ζ such that $|\zeta|$ is minimal. Therefore, by definition of consequence, we deduce that $\zeta = \mathsf{g}(\zeta_1, \ldots, \zeta_n)$ with $\mathsf{g} \in \mathcal{F}_d$ and for all $i \in \{1, \ldots, n\}$, $\zeta_i \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e))$. Since $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$ we also deduce that $\mathsf{g}(\zeta_1, \ldots, \zeta_n)\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$ is a protocol term. Therefore, there exist a rewrite rule $\mathsf{g}(\ell_1, \ldots, \ell_n) \to r$ and a substitution γ such that $\ell_i\gamma = \zeta_i\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$ for all $i = 1 \ldots n$.

Recall that the rule $\textsc{Rew}$ is not applicable on $\mathcal{C}_1^e$ and $\mathcal{C}_2^e$. Therefore we can show if $\neg\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2)$ and $\mathsf{g}(\zeta_1, \ldots, \zeta_n)\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$ is a protocol term then we necessarily have that there exists $\zeta_1', \ldots, \zeta_n'$ and u such that $\mathsf{g}(\zeta_1', \ldots, \zeta_n') \vdash^? u_1 \in \mathsf{F}(\mathcal{C}_1^e)$ and $\zeta_i'\Sigma\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow =$

$\zeta_i\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$. Moreover, since the normalisation rules are also not applicable (in particular rule 19), we deduce that there exists u_2 such that $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n) \vdash^? u_2 \in \mathsf{F}(\mathcal{C}_2^e)$. By $\mathcal{P}_{form}(\mathcal{C}_1^e)$, we know that for all $i \in \{1,\ldots,n\}$, $\zeta'_i \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e) \cup \mathsf{D}(\mathcal{C}_1^e))$ and so $\zeta'_i\Sigma \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$. Moreover, by hypothesis on ζ_i, we know that $\zeta_i \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$. Thus, by applying our inductive hypothesis, we obtain that $\zeta_i\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow = \zeta'_i\Sigma\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow$. Moreover, by $\mathcal{P}_{sound}(\mathcal{C}_2^e)$, we know that $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n)\Sigma\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow = u_2\sigma_2$ which is a protocol term. We can conclude that $\mathsf{g}(\zeta_1,\ldots,\zeta_n)\Sigma\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow$ is a protocol term and so $\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2)$ giving us a contradiction.

Assume now that $\xi\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow = \xi'\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$, $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$ and $\mathit{Msg}(\xi'\Phi(\mathcal{C}_1^e)\sigma_1)$. Let us once again take the smallest $\zeta \in \mathit{st}(\xi,\xi')$ such that $\zeta \not\in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)$. We already proved above that there exist u_1, u_2, g, $\zeta'_1,\ldots,\zeta'_n,\zeta_1,\ldots,\zeta_n$ such that:

- $\zeta = \mathsf{g}(\zeta_1,\ldots,\zeta_n)$
- $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n) \vdash^? u_1 \in \mathsf{F}(\mathcal{C}_1^e)$
- $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n) \vdash^? u_2 \in \mathsf{F}(\mathcal{C}_2^e)$
- for all $i \in \{1,\ldots,n\}$, $\zeta_i\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow = \zeta'_i\Sigma\Phi(\mathcal{C}_2^e)\sigma_2\!\downarrow$ and $\zeta_i\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow = \zeta'_i\Sigma\Phi(\mathcal{C}_1^e)\sigma_1\!\downarrow$.

By $\mathcal{P}_{cons}(\mathcal{C}_1^e)$, we know that there exists β such that $(\beta, u_1) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e) \cup \mathsf{D}(\mathcal{C}_1^e)$. However the normalisation rule 21 is not applicable on the set of extended symbolic processes. Thus, we deduce that there exists β' such that $(\beta', u_1) \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e) \cup \mathsf{D}(\mathcal{C}_1^e)$ and $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \beta' \in \mathsf{F}(\mathcal{C}_1^e)$. Once again due to the normalisation rule 19, we obtain that $\mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \beta' \in \mathsf{F}(\mathcal{C}_2^e)$. But $\mathcal{P}_{sound}(\mathcal{C}_2^e)$ and $\mathcal{P}_{sound}(\mathcal{C}_1^e)$ hold meaning that $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \beta'$ and $(\Phi(\mathcal{C}_1^e)\sigma_1, \Sigma, \sigma_1) \models \mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \beta'$.

Note that if p is the position of ζ in ξ then we have $N(\xi[\beta'\Sigma]_p, \xi') < N(\xi,\xi')$. Thus by applying our inductive hypothesis, we obtain that $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \xi[\beta'\Sigma]_p =^?_f \xi'$. Since $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \beta'$ and $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \mathsf{g}(\zeta'_1,\ldots,\zeta'_n) =^?_f \mathsf{g}(\zeta_1,\ldots,\zeta_n)$, we conclude that $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \xi =^?_f \xi'$.

Inductive step $(0, \max(|\xi|, \xi'|)$ *with* $\max(|\xi|, \xi'| > 0$*:* In such a case, we know that $\xi, \xi' \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_1^e)\Sigma)$ and $\xi, \xi' \in \textsc{Conseq}(\mathsf{K}(\mathcal{C}_2^e)\Sigma)$. By definition of consequence and by $\mathcal{P}_{sound}(\mathcal{C}_1^e)$ and $\mathcal{P}_{sound}(\mathcal{C}_2^e)$, we directly obtain that $\mathit{Msg}(\xi\Phi(\mathcal{C}_1^e)\sigma_1)$ and $\mathit{Msg}(\xi\Phi(\mathcal{C}_2^e)\sigma_2)$ (same thing for ξ'). Now assume that $(\Phi(\mathcal{C}_1^e)\sigma_1, \Sigma, \sigma_1) \models \xi =^?_f \xi'$. Since both ξ, ξ' are consequence of $\mathsf{K}(\mathcal{C}_1^e)\Sigma$, we deduce that:

- either $\xi = \mathsf{f}(\xi_1,\ldots,\xi_n)$ and $\xi' = \mathsf{f}(\xi'_1,\ldots,\xi'_n)$ with $\mathsf{f} \in \mathcal{F}_c$ and $(\Phi(\mathcal{C}_1^e)\sigma_1, \Sigma, \sigma_1) \models \xi_i =^?_f \xi'_i$ for all i. Therefore, we can apply our inductive hypothesis on the (ξ_i, ξ'_i)s to conclude.
- or $\xi\Sigma, \xi'\Sigma \in \mathsf{K}(\mathcal{C}_1^e)\Sigma$: Since we know that the rule Equality is not applicable, it implies that $\xi =^?_f \xi' \in \mathsf{F}(\mathcal{C}_1^e)$ and so $\xi =^?_f \xi' \in \mathsf{F}(\mathcal{C}_2^e)$ thanks to the normalisation rule 19. Since $\mathcal{P}_{sound}(\mathcal{C}_2^e)$ holds, we can conclude that $(\Phi(\mathcal{C}_2^e)\sigma_2, \Sigma, \sigma_2) \models \xi =^?_f \xi'$.
- or $\xi\Sigma \in \mathsf{K}(\mathcal{C}_1^e)\Sigma$ and $\xi' = \mathsf{f}(\xi'_1,\ldots,\xi'_n)$ with $\mathsf{f} \in \mathcal{F}_c$; Once again since the rule Equality is not applicable, we deduce that there exists $\zeta'_1,\ldots,\zeta'_n$ such that $\xi =^?_f \mathsf{f}(\zeta'_1,\ldots,\zeta'_n) \in \mathsf{F}(\mathcal{C}_1^e)$. Note from $\mathcal{P}_{sound}(\mathcal{C}_1^e)$ that in such a case, $(\Phi(\mathcal{C}_1^e)\sigma_1, \Sigma, \sigma_1) \models$

$\xi =^?_f \mathsf{f}(\zeta'_1, \ldots, \zeta'_n)$ meaning that $(\Phi(\mathcal{C}^e_1)\sigma_1, \Sigma, \sigma_1) \models \xi'_i =^?_f \zeta'_i$ for all $i \in \{1, \ldots, n\}$. Since $|\xi'_i\Phi(\mathcal{C}^e_1)\sigma_1{\downarrow}| < |\xi\Phi(\mathcal{C}^e_1)\sigma_1{\downarrow}|$, we can apply our inductive hypothesis on all (ξ'_i, ζ'_i) meaning that $(\Phi(\mathcal{C}^e_2)\sigma_2, \Sigma, \sigma_2) \models \mathsf{f}(\zeta'_1, \ldots, \zeta'_n) =^?_f \xi'$. However, thanks to the normalisation rule 19 not being applicable, $\xi =^?_f \mathsf{f}(\zeta_1, \ldots, \zeta'_n) \in \mathsf{F}(\mathcal{C}^e_1)$ implies $\xi =^?_f \mathsf{f}(\zeta'_1, \ldots, \zeta'_n) \in \mathsf{F}(\mathcal{C}^e_2)$ and so by $\mathcal{P}_{sound}(\mathcal{C}^e_2)$, we obtain that $(\Phi(\mathcal{C}^e_2)\sigma_2, \Sigma, \sigma_2) \models \xi =^?_f \mathsf{f}(\zeta'_1, \ldots, \zeta'_n)$ which allows us to conclude that $(\Phi(\mathcal{C}^e_2)\sigma_2, \Sigma, \sigma_2) \models \xi =^?_f \xi'$. $\square$