Exploiting symmetries when proving equivalence properties for security protocols (Technical report)

Vincent Cheval, Steve Kremer, Itsaka Rakotonirina

Exploiting symmetries when proving equivalence properties for security protocols (Technical report[1])

Vincent Cheval, Steve Kremer, Itsaka Rakotonirina
INRIA Nancy Grand-Est & LORIA

ABSTRACT

Verification of privacy-type properties for cryptographic protocols in an active adversarial environment, modelled as a behavioural equivalence in concurrent-process calculi, exhibits a high computational complexity. While undecidable in general, for some classes of common cryptographic primitives the problem is coNEXP-complete when the number of honest participants is bounded.

In this paper we develop optimisation techniques for verifying equivalences, exploiting symmetries between the two processes under study. We demonstrate that they provide a significant (several orders of magnitude) speed-up in practice, thus increasing the size of the protocols that can be analysed fully automatically.

1 INTRODUCTION

Security protocols are distributed programs transmitting data between several parties. The underlying messages may be sensitive—for economical, political, or privacy reasons—and communications are usually performed through an untrusted network such as the Internet. Therefore, such protocols need to guarantee strong security requirements in an *active* adversarial setting, i.e., when considering an adversary that has complete control over the communication network. Formal, symbolic methods, rooted in the seminal work of Dolev and Yao [DY81], have been successful in analysing complex protocols, including for instance the recent TLS 1.3 proposal [BBK17, CHH+17] and the upcoming 5G standard [BDH+18b].

While some security properties can be formalised as reachability statements, privacy related properties are generally defined as the indistinguishability of two situations where the value of a private attribute differs. This is why privacy-type properties such as anonymity, (strong flavors of) secrecy, unlinkability, or privacy in e-voting are often modelled as behavioural equivalences in concurrent process calculi, such as the applied pi-calculus [ABF18]. The problem of verifying such equivalences is undecidable in the full, Turing-complete, calculus. Still, decidability results and fully automated analysers exist when the number of protocol sessions is bounded.

Unfortunately, recent results [CKR18a] show that the problem has a high computational worst-case complexity (coNEXP-complete). Yet, other results show that the problem is exponentially simpler (coNP-complete) for a class of practical scenarios (*determinate processes*) [CD09]. This gap is all the more striking in practice as, for determinate processes, the verification time can effectively be reduced by several orders of magnitude using *partial-order reductions* [BDH15, CKR18b]. This highlights the gap between the general, pessimistic complexity bound and what can be achieved by exploiting specificities of given instances. In practice, the processes that are analysed show a great amount of symmetries as they often consist of several copies (sessions) of the same protocol executed in parallel. Exploiting this helps factoring out large, redundant parts of equivalence proofs, and making theoretically hard verification feasible in practice.

Contributions

We present optimisations for the verification of trace equivalence in the applied pi-calculus. For that we exploit the symmetries of the two processes to be shown equivalent. More specifically, our contributions are as follows.

(1) We introduce *equivalence by session*, a new process equivalence that implies the classical trace equivalence. Intuitively, it is a refinement of trace equivalence designed for two processes sharing a similar structure, making verification easier.
(2) We show how partial-order reductions, some of which being presented in [BDH15] for determinate processes, can be used for proving equivalence by session for *any* processes.
(3) We give a group-theoretic characterisation of internal process redundancy, inspired by classical formalisations of symmetries in model checking [ES96], and use it to reduce further the complexity of deciding equivalence by session.
(4) We design a symbolic version of the above equivalence and optimisations, based on the constraint solving techniques of the DeepSec prover [CKR18b], a state-of-the-art tool for verifying equivalence properties in security protocols. This allowed us to implement our techniques in DeepSec and evaluate the gain in verification time induced by our optimisations.

Note that, while we designed equivalence by session as an efficient proof technique for trace equivalence it is also of independent interest: to some extent, equivalence by session models attackers that can distinguish different sessions of a same protocol. This may be considered realistic when servers allocate a distinct ephemeral port for each session; in other contexts, e.g. RFID communication this may however be too strong. When equivalence by session is used as a proof technique for trace equivalence, false attacks are possible, as it is a sound, but not complete, refinement. However, on the existing protocols we experimented on, when equivalence by session was violated, trace equivalence was violated as well.

[1] This is the technical report of the conference paper [CKR19]. It contains technical proofs and generalised results, including the features of the DeepSec 2.0 release (Jan. 2020). Significant experimental improvements can thus be observed compared to [CKR19], mostly due to the new optimisations and a more efficient implementation of the data structures. We also provide a different running example for a complementary presentation.

Our prototype is able to successfully analyse various security protocols that are currently out of scope—in terms of expressivity or exceeding a 12h timeout—of similar state-of-the-art analysers. We observe improvements of several orders of magnitude in terms of efficiency, compared to the original version of DeepSec. Among the case studies that we consider are

- the *Basic Acces Control (BAC)* protocol [For04] implemented in European e-passports. In previous work, verification was limited to merely 2 sessions, while we scale up to 5 sessions.
- the *Helios* e-voting protocol [Adi08]. Automated analyses of this protocol exist when no revote is allowed, or is limited to one revote from a honest voter [ACK16, CKR18a]. In this paper, we analyse several models covering revote scenarios for 7 emitted honest ballots.

Related work

Partial-order reductions (por) for the verification of cryptographic protocols were first introduced by Clark et al. [CJM03]: while well developed in verification of reactive systems these existing techniques do not easily carry over to security protocols, mainly due to the symbolic treatment of attacker knowledge. Mödersheim et al. [MVB10] proposed por techniques that are suitable for symbolic methods based on constraint solving. However, both the techniques of Clark et al. [CJM03] and Mödersheim et al. [MVB10] are only correct for trace properties.

Partial order reductions for equivalence properties were only introduced more recently by Baelde et al. [BDH14, BDH15]: implementing these techniques in the APTE tool resulted in spectacular speed-ups. Other state-of-the-art tools, AKISS [CCCK16] and DeepSec [CKR18a], integrated these techniques as well. However, these existing techniques are limited in scope as they require protocols to be *determinate*. Examples of protocols that are typically not modelled as determinate processes are the BAC protocol, and the Helios e-voting protocol mentioned above. In recent work, Baelde et al. [BDH18a] propose por techniques that also apply to non-determinate processes (but do not support private channels) and implement these techniques in the DeepSec tool. Unfortunately, these techniques introduce a computational overhead, that tends to limit the efficiency gain. As our experiments will show, our techniques, although including some approximations, significantly improve efficiency.

There exist other tools for the verification of equivalence properties in the case of a bounded number of sessions. The SAT-Equiv tool [CDD17] is extremely efficient, but its scope is more narrow: it does not support user-defined equational theories and is restricted to determinate processes. As shown in [CKR18a], AKISS [CCCK16] and SPEC [TNH16] were already less efficient (by orders of magnitude) than DeepSec before our current work. We also mention the less recent S^3A tool [DSV03] that verifies testing equivalence in the SPI calculus and integrates some symmetry (but no partial-order) reductions [CDSV04]. The tool however only supports a fixed equational theory and no else branches. We are not aware of a publicly available implementation.

Our approach can also be compared to tools for an unbounded number of sessions. The ProVerif [BAF08], Tamarin [BDS15] and Maude-NPA [SEMM14] tools all show a process equivalence that is more fine-grained than trace equivalence. The resulting equivalence is often referred to as *diff-equivalence* in that it requires that equivalent processes follow the same execution flow and only differ on the data. As a result these techniques may fail to prove equivalence of processes that are trace equivalent. Our approach goes in the same direction but equivalence by session is less fine-grained, for example capturing equivalence proofs for the BAC protocol. A detailed comparison between these two equivalences is given in Section 3.1. Besides, the restriction to a bounded number of sessions allows us to decide equivalence by session, while termination is not guaranteed in the unbounded case.

2 MODEL

We first present our model for formalising privacy-type properties of security protocols, represented by trace equivalence of processes in the applied-pi calculus [ABF18].

2.1 Messages and cryptography

In order to analyse protocols, we rely on symbolic models rooted in the seminal work of Dolev and Yao [DY81]. Cryptographic operations are modelled by a finite *signature*, i.e., a set of function symbols with their arity $\mathcal{F} = \{\mathsf{f}/n, \mathsf{g}/m, \ldots\}$. Atomic data such as nonces, random numbers, or cryptographic keys are represented by an infinite set of *names*

$$\mathcal{N} = \{a, b, k, \ldots\} = \mathcal{N}_{pub} \cup \mathcal{N}_{priv}$$

partitioned into *public* and *private* names. We also consider an infinite set of variables $\mathcal{X} = \{x, y, z, \ldots\}$. Protocol messages are then modelled as terms obtained by application of function symbols to names, variables or other terms. If $A \subseteq \mathcal{N} \cup \mathcal{X}$, $\mathcal{T}(\mathcal{F}, A)$ refers to the set of terms built from atoms in A.

Example 2.1. The following signature models the classical primitives of pairs, randomised symmetric encryption, and their inverse:

$$\mathcal{F} = \{\ \langle\cdot,\cdot\rangle/2,\ \mathsf{proj}_1/1,\ \mathsf{proj}_2/1,\ \mathsf{senc}/3,\ \mathsf{sdec}/2\ \}$$

For example, let $m \in \mathcal{N}_{pub}$, and $k, r \in \mathcal{N}_{priv}$ modelling a private key and a random nonce, respectively. The term

$$c = \mathsf{senc}(m, r, k)$$

models a ciphertext obtained by encrypting m with the key k and randomness r, and $\mathsf{sdec}(c, k)$ models its decryption. △

An *equational theory* is a binary relation E on terms. It is extended to an equivalence relation $=_E$ that is the closure of E by reflexivity, symmetry, transitivity, substitution and applications of function symbols. All the optimisations we present in this paper are sound for arbitrary equational theories although, obviously, the implementation in DeepSec naturally inherits the restrictions of the tool (limited to destructor subterm convergent rewriting systems). The following equations characterise the behaviour of the primitives introduced in Example 2.1:

$$\mathsf{proj}_i(\langle x_1, x_2\rangle) =_E x_i \qquad \mathsf{sdec}(\mathsf{senc}(x, y, z), z) =_E x$$

That is, a message encrypted with a key k can be recovered by decrypting using the same key k. With the notations of Example 2.1, we can derive from these equations that $\mathsf{sdec}(c, k) =_E m$.

A *substitution* σ is a mapping from variables to terms, homomorphically extended to a function from terms to terms. We use the classical postfix notation $t\sigma$ instead of $\sigma(t)$, and the set notation $\sigma = \{x_1 \mapsto x_1\sigma, \ldots, x_n \mapsto x_n\sigma\}$.

2.2 Protocols as processes

Syntax Protocols are modelled as concurrent processes that exchange messages (i.e. terms). We define the syntax of *plain processes* by the following grammar:

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

where u, v are terms, $x \in \mathcal{X}$, and $c \in Ch$ where Ch denotes a set of *channels*. We assume a partition $Ch = Ch_{pub} \cup Ch_{priv}$ of channels into public and private channels: while public channels are under the control of the adversary, private channels allow confidential, internal communications. The 0 process is the terminal process which does nothing, the operator $P \mid Q$ executes P and Q concurrently, $\overline{c}\langle u\rangle$ sends a message u on channel c, and $c(x)$ receives a message (and binds it to the variable x).

We highlight the two restrictions compared to the calculus of [ABF18]: we only consider a bounded number of protocol sessions (i.e. there is no operator for unbounded parallel replication) and channels are modelled by a separate datatype (i.e. they are never used as parts of messages). The first restriction is necessary for decidability [CCCK16, TNH16, CKR18a] but still allows to detect many flaws since attacks tend to require a rather small number of sessions. Our optimisations also rely on an invariant that private channels remain unknown to the adversary, hence the restriction to disallow channel names in messages.

Example 2.2. We describe a toy protocol that will serve as a running example throughout the paper. This is a simplification of the BAC protocol implemented in the European e-passports. The system builds upon the signature and equational theory introduced in Example 2.1. In a preliminary phase, a reader obtains the private key k of a passport, and then they communicate as follows:

$$\begin{array}{rcll} \textit{Reader} \longrightarrow \textit{Passport} & : & \textsc{get\_challenge} & \\ \textit{Passport} \longrightarrow \textit{Reader} & : & n & \\ \textit{Reader} \longrightarrow \textit{Passport} & : & \mathrm{senc}(n, r, k) & \text{bound as } x \\ \textit{Passport} \longrightarrow \textit{Reader} & : & \textsc{ok} & \text{if } \mathrm{sdec}(x, k) = n \\ & & \textsc{error} & \text{otherwise} \end{array}$$

where $n, r \in \mathcal{N}_{priv}$ and $\textsc{get\_challenge}, \textsc{ok}, \textsc{error} \in \mathcal{N}_{pub}$. In particular, the passport triggers an error when it receives a communication originated from a reader that has not the right key k, i.e. a reader that has been paired with an other passport during the preliminary phase. In the applied pi-calculus, they are modelled by the following processes

$$\begin{aligned} R(k, r) = {} & \overline{c}\langle \textsc{get\_challenge}\rangle. \\ & c(x_n).\, \overline{c}\langle \mathrm{senc}(x_n, r, k)\rangle.\, 0 \end{aligned}$$

$$\begin{aligned} P(k, n) = {} & c(x_0).\ \text{if } x_0 = \textsc{get\_challenge} \text{ then} \\ & \quad \overline{c}\langle n\rangle.\, c(x). \\ & \quad \text{if } \mathrm{sdec}(x, k) = n \text{ then } \overline{c}\langle \textsc{ok}\rangle.\, 0 \\ & \quad \text{else } \overline{c}\langle \textsc{error}\rangle.\, 0 \end{aligned}$$

△

Semantics The behaviour of protocols is defined by an operational semantics on processes. Its first ingredients are simplifying rules to normalise processes from non-observable, deterministic actions (Figure 1).

$$P \mid 0 \rightsquigarrow P \qquad 0 \mid P \rightsquigarrow P \qquad (P \mid Q) \mid R \rightsquigarrow P \mid (Q \mid R)$$

$$\left.\begin{array}{l} P \mid Q \rightsquigarrow P' \mid Q \\ Q \mid P \rightsquigarrow Q \mid P' \end{array}\right\} \text{ if } P \rightsquigarrow P'$$

$$\text{if } u = v \text{ then } P \text{ else } Q \rightsquigarrow \left\{\begin{array}{ll} P & \text{if } u =_E v \\ Q & \text{otherwise} \end{array}\right.$$

Figure 1: Simplification rules for plain processes

These simplifying rules get rid of 0 processes, and evaluate conditionals at toplevel. We say that a process on which no more rule applies is in $\rightsquigarrow$-normal form. By convergence, we will denote by $P{\wr}$ the unique $\rightsquigarrow$-normal form of P.

The operational semantics then operates on *extended processes* $(\mathcal{P}, \Phi)$, where $\mathcal{P}$ is a multiset of plain processes (in $\rightsquigarrow$-normal form) and Φ is a substitution, called the *frame*. Intuitively, $\mathcal{P}$ is the multiset of processes that are ready to be executed in parallel, and Φ is used to record outputs on public channels. The domain of the substitution Φ is a subset of a set $\mathcal{AX}$ of *axioms*, disjoint from $\mathcal{X}$: they record the raw observations of the attacker, that is, they are the axioms in intruder deduction proofs. The semantics (Figure 2) takes the form of a labelled transition relation $\xrightarrow{\alpha}$ between extended processes, where α is called an *action* and indicates what kind of transition is performed.

The output rule (Out) models that outputs on a public channel are added to the attacker knowledge, i.e., stored in Φ in a fresh axiom. The axioms thus provide handles for the attacker to refer to these outputs. The input rule (In) reads a term ξ, called a *recipe* provided by the attacker, on a public channel. This term ξ can be effectively constructed by the attacker as it is built over public names and elements of $dom(\Phi)$, i.e. previous outputs. The resulting term is then bound to the input variable x. Rule (Comm) models internal communication on a private channel and rule (Par) adds processes in parallel to the multiset of active processes. These last two actions are internal actions (label τ), unobservable by the attacker.

Traces A *trace* of an extended process A is a sequence of reduction steps starting from the extended process A

$$t : A \xrightarrow{\alpha_1} A_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} A_n\,.$$

When the intermediate processes are not relevant we write

$$t : A \xRightarrow{\alpha_1 \cdots \alpha_n} A_n\,.$$

$$
\begin{array}{llcll}
(\text{Out}) & (\{\!\{\bar{c}\langle u\rangle.P\}\!\} \cup \mathcal{P}, \Phi) & \xrightarrow{\bar{c}\langle \mathsf{ax}\rangle} & (\{\!\{P\}\!\} \cup \mathcal{P}, \Phi \cup \{\mathsf{ax} \mapsto u\}) & c \in Ch_{pub}, \mathsf{ax} \in \mathcal{AX} \setminus dom(\Phi) \\
(\text{In}) & (\{\!\{c(x).P\}\!\} \cup \mathcal{P}, \Phi) & \xrightarrow{c(\xi)} & (\{\!\{P[x \mapsto \xi\Phi]\!\downarrow\}\!\} \cup \mathcal{P}, \Phi) & c \in Ch_{pub}, \xi \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{pub} \cup dom(\Phi)) \\
(\text{Comm}) & (\{\!\{\bar{c}\langle u\rangle.P, c(x).Q\}\!\} \cup \mathcal{P}, \Phi) & \xrightarrow{\tau} & (\{\!\{P, Q[x \mapsto u]\}\!\} \cup \mathcal{P}, \Phi) & c \in Ch_{priv} \\
(\text{Par}) & (\{\!\{P_1 \mid \ldots \mid P_n\}\!\} \cup \mathcal{P}, \Phi) & \xrightarrow{\tau} & (\{\!\{P_1, \ldots, P_n\}\!\} \cup \mathcal{P}, \Phi) &
\end{array}
$$

Figure 2: Operational semantics of the applied pi-calculus

We define $tr(t)$ to be the word of actions $\alpha_1 \cdots \alpha_n$ (including τ's), and $\Phi(t)$ to be the frame of A_n. The set of the traces of A is written $\mathbb{T}(A)$, and the notation is extended to plain processes by writing $\mathbb{T}(P)$ for $\mathbb{T}(\{\!\{P\}\!\}, \emptyset)$.

Example 2.3. Consider again the access-control protocol described in Example 2.2. Let $S = (\{\!\{P(k,n), R(k',r)\}\!\},\ \emptyset)$, with $k, k', n, r \in \mathcal{N}_{priv}$, a system consisting of a passport and a reader in parallel. The system has the following trace:

$$
\begin{aligned}
S &\xrightarrow{\bar{c}\langle \mathsf{ax}_0\rangle} (\{\!\{P(k,n), R_0(k',r)\}\!\}, \Phi_0) \\
&\xrightarrow{c(\textsc{get\_challenge})} (\{\!\{P_0(k,n), R_0(k',r)\}\!\},\ \Phi_0) \\
&\xrightarrow{\bar{c}\langle \mathsf{ax}_1\rangle} (\{\!\{P_1(k,n), R_0(k',r)\}\!\},\ \Phi_0 \cup \Phi_1) \\
&\xrightarrow{c(\mathsf{ax}_1)} (\{\!\{P_1(k,n), \bar{c}\langle \mathsf{senc}(n,r,k')\rangle\}\!\},\ \Phi_0 \cup \Phi_1) \\
&\xrightarrow{\bar{c}\langle \mathsf{ax}_2\rangle} (\{\!\{P_1(k,n), 0\}\!\},\ \Phi_0 \cup \Phi_1 \cup \Phi_2) \\
&\xrightarrow{c(\mathsf{ax}_2)} (\{\!\{\bar{c}\langle \alpha\rangle, 0\}\!\},\ \Phi_0 \cup \Phi_1 \cup \Phi_2)\})
\end{aligned}
$$

with

$$
\begin{aligned}
\Phi_0 &= \{\mathsf{ax}_0 \mapsto \textsc{get\_challenge}\} \\
\Phi_1 &= \{\mathsf{ax}_1 \mapsto n\} \\
\Phi_2 &= \{\mathsf{ax}_2 \mapsto \mathsf{senc}(n,r,k')\} \\
P(k,n) &= c(x_0).\ \mathsf{if}\ x_0 = \textsc{get\_challenge}\ \mathsf{then}\ P_0(k,n) \\
P_0(k,n) &= \bar{c}\langle n\rangle.\ P_1(k,n) \\
R(k',r) &= \bar{c}\langle \textsc{get\_challenge}\rangle.\ R_0(k',r)
\end{aligned}
$$

and $\alpha = \textsc{ok}$ if $k = k'$, and $\alpha = \textsc{error}$ if $k \neq k'$. Note that the input action $c(\textsc{get\_challenge})$ could be replaced by $c(\mathsf{ax}_0)$. △

2.3 Security properties

Many security properties can be expressed in terms of indistinguishability (from the attacker's viewpoint). The preservation of anonymity during a protocol execution can for example be modelled as the indistinguishability of two instances of the protocol with different participants. Strong flavors of secrecy can also be expressed: after interacting with the protocol, the attacker is still unable to distinguish between a secret used during the protocol and a fresh random nonce. Our case studies also include such modellings of unlinkability or vote privacy.

Static equivalence The ability to distinguish or not between two situations lies on the attacker's observations, i.e. the frame. Indistinguishability of two frames is captured by the notion of *static equivalence*. Intuitively, we say that two frames are statically equivalent if the attacker cannot craft an equality test that holds in one frame and not in the other.

Definition 2.1. Two frames Φ_1 and Φ_2 are statically equivalent, written $\Phi_1 \sim \Phi_2$ when $dom(\Phi_1) = dom(\Phi_2)$ and, for any recipes $\xi_1, \xi_2 \in \mathcal{T}(\mathcal{F}, \mathcal{N}_{pub} \cup dom(\Phi_1))$,

$$\xi_1\Phi_1 =_E \xi_2\Phi_1 \quad \Leftrightarrow \quad \xi_1\Phi_2 =_E \xi_2\Phi_2$$

We lift static equivalence to traces and write $t_0 \sim t_1$ when $\Phi(t_0) \sim \Phi(t_1)$ and $\mathsf{tr}_0 = \mathsf{tr}_1$, where tr_i is obtained by removing τ's from $tr(t_i)$. Removing τ actions reflects that these actions are unobservable by the attacker.

Trace equivalence While static equivalence models the (passive) indistinguishability of two sequences of observations, *trace equivalence* captures the indistinguishability of two processes P and Q in the presence of an active attacker. Intuitively, we require that any sequence of visible actions executable on P is also executable on Q and yields indistinguishable outputs, i.e., statically equivalent frames.

Definition 2.2. Let P, Q be plain processes in $\rightsquigarrow$-normal form. P is *trace included* in Q, written $P \sqsubseteq_{tr} Q$, when

$$\forall t \in \mathbb{T}(P),\ \exists t' \in \mathbb{T}(Q),\ t \sim t'\,.$$

We say that P and Q are *trace equivalent*, written $P \approx_{tr} Q$, when $P \sqsubseteq_{tr} Q$ and $Q \sqsubseteq_{tr} P$.

Example 2.4. Consider again the model of passport and reader introduced in Example 2.2, and let us write

$$S(k,n,r) = P(k,n) \mid R(k,r)$$

a system consisting of a passport and a reader in parallel with a shared key k. The unlinkability property can be stated by the inability for the attacker to distinguish between two copies of the same passport interacting with readers, and two different passports. That is,

$$S(k,n,r) \mid S(k,n',r') \approx_{tr} S(k,n,r) \mid S(k',n',r')$$

with $k, k', n, n', r, r' \in \mathcal{N}_{priv}$ pairwise distinct. The inclusion $S(k,n,r) \mid S(k,n',r') \sqsubseteq_{tr} S(k,n,r) \mid S(k',n',r')$ indeed holds. However, this model of unlinkability is violated in that the converse inclusion does not hold. Indeed in the right-hand-side process, making a reader interact with the wrong passport produces an error message, which is not the case in the left-hand side (since the two systems share the same key). Formally, $\mathbb{T}(S(k, n, r) \mid S(k', n', r'))$ contains a trace t such that

$$tr(t) = \tau\ \overline{c}\langle \mathsf{ax}_0\rangle\ c(\mathsf{ax}_0)\ \overline{c}\langle \mathsf{ax}_1\rangle\ c(\mathsf{ax}_1)\ \overline{c}\langle \mathsf{ax}_2\rangle\ c(\mathsf{ax}_2)\ \overline{c}\langle \mathsf{ax}_3\rangle$$
$$\Phi(t) = \{\mathsf{ax}_0 \mapsto \textsc{get\_challenge},\ \mathsf{ax}_1 \mapsto n,\ \mathsf{ax}_2 \mapsto \mathsf{senc}(n, r', k'),\ \mathsf{ax}_3 \mapsto \textsc{error}\} \qquad \triangle$$

3 OPTIMISING VERIFICATION

The problem of verifying trace equivalence in the presented model is coNEXP-complete for equational theories represented as subterm convergent destructor rewrite systems [CKR18a]. Despite this high theoretical complexity, automated analysers can take advantage of the specificities of practical instances. One notable example is the class of *determinate* processes that encompasses many practical scenarios and has received quite some attention [CD09, BDH15, CCCK16, CKR18b]. It allows for partial-order reductions [BDH15], speeding up the verification time by several orders of magnitude. Our approach, similar in spirit but applicable in a more general setting, consists in guiding the decision procedure with the structural similarities of the two processes that we aim to show equivalent.

3.1 Equivalence by session

We introduce a new equivalence relation, *equivalence by session*: the main idea is that, when proving the equivalence of P and Q, every action of a given parallel subprocess of P should be matched by the actions of a same subprocess in Q. This is indeed often the case in protocol analysis where a given session (the execution of an instance of a protocol role) on one side is matched by a session on the other side. By requiring to match sessions rather than individual actions, this yields a more fine-grained equivalence and effectively reduces the combinatorial explosion. Moreover, thanks to the optimisations that exploit the structural properties of equivalence by session (presented in the following sections), we obtain significant speed-ups during the verification of case studies that are neither determinate nor in scope of the (even more fined-grained) diff-equivalence of ProVerif and Tamarin.

Twin processes To formalise session matchings we use a notion of *twin-process*, that are pairs of matched processes that have the same action at toplevel, called their *skeleton*.

Definition 3.1. A twin-process is a pair of plain processes in $\rightsquigarrow$-normal form (P, Q) such that $\mathsf{skel}(P) = \mathsf{skel}(Q)$, where

$$\text{if } c \in Ch_{pub}\text{:}\quad \mathsf{skel}(c(x).Q) = \{\!\!\{\mathsf{in}_c\}\!\!\} \quad \mathsf{skel}(\overline{c}\langle x\rangle.Q) = \{\!\!\{\mathsf{out}_c\}\!\!\}$$
$$\text{if } d \in Ch_{priv}\text{:}\quad \mathsf{skel}(d(x).Q) = \{\!\!\{\mathsf{in}\}\!\!\} \quad \mathsf{skel}(\overline{d}\langle x\rangle.Q) = \{\!\!\{\mathsf{out}\}\!\!\}$$
$$\mathsf{skel}(P_1 \mid \cdots \mid P_n) = \mathsf{skel}(P_1) \cup \ldots \cup \mathsf{skel}(P_n)$$

An *extended twin-process* $A^2 = (\mathcal{P}^2, \Phi_0, \Phi_1)$ is then a triple where $\mathcal{P}^2$ is a multiset of twin-processes and Φ_0, Φ_1 are frames. This thus models two extended processes with identical skeletons, matched together. We retrieve the original extended processes by projection,

$$\mathsf{fst}(A^2) = (\{\!\!\{P_0 \mid (P_0, P_1) \in \mathcal{P}^2\}\!\!\}, \Phi_0)$$
$$\mathsf{snd}(A^2) = (\{\!\!\{P_1 \mid (P_0, P_1) \in \mathcal{P}^2\}\!\!\}, \Phi_1)$$

The semantics of twin-processes is defined in Figure 3 and mostly requires that the two projections follow the same reduction steps in the single-process semantics. The rule (Par) is however replaced by a rule that allows to match each parallel subprocess from the left with a parallel process from the right. We underline that, by definition of twin-processes, a transition $A^2 \xrightarrow{\alpha}_s (\mathcal{P}^2, \Phi)$ is possible only if for all $(P, Q) \in \mathcal{P}^2$, it holds that $\mathsf{skel}(P) = \mathsf{skel}(Q)$.

Similarly to extended processes, we use $\mathbb{T}(A^2)$ to denote the set of reduction steps from an extended twin-process A^2. Besides if

$$t^2 : A^2 \xrightarrow{\alpha_1}_s A_1^2 \cdots \xrightarrow{\alpha_n}_s A_n^2 \quad \in \mathbb{T}(A_1^2),$$

we also lift the projection functions by writing

$$\mathsf{fst}(t^2) : \mathsf{fst}(A^2) \xrightarrow{\alpha_1}_s \mathsf{fst}(A_1^2) \cdots \xrightarrow{\alpha_n}_s \mathsf{fst}(A_{n+1}^2)$$

and similarly for $\mathsf{snd}(t^2)$. Note that $\mathsf{fst}(t^2) \in \mathbb{T}(\mathsf{fst}(A^2))$.

Equivalence by session Equivalence by session is similar to trace equivalence but only considers the traces of Q matching the structure of the trace of P under study. This structural requirement is formalised by considering traces of the twin-process (P, Q). Formally speaking, given two plain processes P and Q in $\rightsquigarrow$-normal form having the same skeleton, we write $P \sqsubseteq_s Q$ when

$$\forall t \in \mathbb{T}(P),\ \exists t^2 \in \mathbb{T}(P, Q),\ t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)\,.$$

We say that P and Q are *equivalent by session*, referred as $P \approx_s Q$, when $P \sqsubseteq_s Q$ and $Q \sqsubseteq_s P$.

While equivalence by session has been designed to increase efficiency of verification procedures, it is also of independent interest. Equivalence by session captures a notion of indistinguishability against an adversary that is able to distinguish actions which originate from different protocol sessions. Such an adversarial model may for instance be considered realistic in protocols where servers dynamically allocate a distinct *ephemeral port* to each session. An attacker would therefore observe these ports and always differentiate one session from another. When considering equivalence by session, this allocation mechanism does not need to be explicitly modelled as it is already reflected natively in the definition. On the contrary for trace equivalence, an explicit modelling within the processes would be needed. For example equivalence by session of two protocol sessions operating on a public channel c,

$$P(c) \mid P(c) \approx_s Q(c) \mid Q(c)$$

could be encoded by relying on dynamically-generated private channels that are revealed to the attacker. This can be expressed in the original syntax of the applied pi-calclulus [ABF18] as:

$$P_{fresh} \mid P_{fresh} \approx_{tr} Q_{fresh} \mid Q_{fresh}$$

where $P_{fresh} = \mathsf{new}\, e.\, \overline{c}\langle e\rangle.\, P(e)$, $Q_{fresh} = \mathsf{new}\, e.\, \overline{c}\langle e\rangle.\, Q(e)$. Such encodings however break determinacy and are thus incompatible with the partial-order reductions of [BDH15]. Our dedicated equivalence offers similar-in-spirit optimisations that are applicable on all processes .

$$\frac{(\{\!\{P_i\}\!\}, \Phi_i) \xrightarrow{\alpha} (\{\!\{P_i'\}\!\}, \Phi_i') \text{ by rule (In) or (Out) for all } i \in \{0,1\}}{(\{\!\{(P_0, P_1)\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1) \xrightarrow{\alpha}_s (\{\!\{(P_0', P_1')\}\!\} \cup \mathcal{P}^2, \Phi_0', \Phi_1')} \quad \text{(IO)}$$

$$\frac{(\{\!\{P_i, Q_i\}\!\}, \Phi_i) \xrightarrow{\tau} (\{\!\{P_i', Q_i'\}\!\}, \Phi_i) \text{ by rule (Comm) for all } i \in \{0,1\}}{(\{\!\{(P_0, P_1), (Q_0, Q_1)\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1) \xrightarrow{\tau}_s (\{\!\{(P_0', P_1'), (Q_0', Q_1')\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1)} \quad \text{(Comm)}$$

$$\frac{\pi \text{ permutation of } [\![1, n]\!]}{(\{\!\{(P_1 \mid \cdots \mid P_n,\ Q_1 \mid \cdots \mid Q_n)\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1) \xrightarrow{\tau}_s (\{\!\{(P_i, Q_{\pi(i)})\}\!\}_{i=1}^n \cup \mathcal{P}^2, \Phi_0, \Phi_1)} \quad \text{(Match)}$$

Figure 3: Semantics on twin-processes

3.2 Comparison to other equivalences

Relation to trace equivalence We first show that equivalence by session is a sound refinement of trace equivalence.

Proposition 3.1 . If $P \approx_s Q$ then $P \approx_{tr} Q$.

This is immediate as $t^2 \in \mathbb{T}(P, Q)$ entails $\mathsf{snd}(t^2) \in \mathbb{T}(Q)$. The converse does not hold in general, meaning that two processes that are not equivalent by session might be trace equivalent. The simplest example is, for $n \in \mathcal{N}_{pub}$,

$$P = \overline{c}\langle n\rangle.\,\overline{c}\langle n\rangle \qquad\qquad Q = \overline{c}\langle n\rangle \mid \overline{c}\langle n\rangle$$

We call *false attacks* traces witnessing a violation of equivalence by session, but that can still be matched trace-equivalence-wise. In this example even the empty trace is a false attack since the two processes fail to meet the requirement of having identical skeletons. Such extreme configurations are however unlikely to occur in practice: privacy is usually modelled as the equivalence of two protocol instances where some private attributes are changed. In particular the overall structure in parallel processes remains common to both sides.

More realistic false attacks may arise when the structural requirements of equivalence by session are too strong, i.e. when matching the trace requires mixing actions from different sessions. Consider for example the two processes

$$P = \overline{s}\langle n\rangle.\,\overline{a}\langle n\rangle \mid s(x).\,\overline{b}\langle n\rangle \qquad Q = \overline{s}\langle n\rangle.\,\overline{b}\langle n\rangle \mid s(x).\,\overline{a}\langle n\rangle$$

with $a, b \in Ch_{pub}$ and $s \in Ch_{priv}$. These processes first synchronise on a private channel s by the means of an internal communication, and then perform two parallel outputs on public channels a, b. They are easily seen trace equivalent. However the skeletons at toplevel constrain the session matchings, i.e. the application of rule (Match). Hence any trace executing an output on a or b is a false attack.

Finally false attacks cannot happen for determinate processes, i.e. the class of processes for which the partial-order reductions of [BDH15] were designed. A plain process P is determinate if it does not contain private channels and,

$$\forall\, P \overset{\mathsf{tr}}{\Longrightarrow} (\{\!\{P_1, \ldots, P_n\}\!\}, \Phi),\ \forall i \neq j,\ \mathsf{skel}(P_i) \neq \mathsf{skel}(P_j)\,.$$

Proposition 3.2 . If P, Q are determinate plain processes such that $P \approx_{tr} Q$ then $P \approx_s Q$.

The core argument is the uniqueness of session matchings; that is, there is always at most one permutation that can be chosen when applying the rule (Match) to a pair of determinate processes. The proof can be found in Appendix B: thanks to the structural requirements imposed by skeletons, we even prove that trace equivalence ($\approx_{tr}$) and inclusion by session ($\sqsubseteq_s$) coincide for determinate processes.

Relation to diff-equivalence ProVerif, Tamarin and Maude-NPA are semi-automated tools that can provide equivalence proofs for an unbounded number of protocol sessions. For that they rely on another refinement of trace equivalence, called diff-equivalence ($\approx_d$). It relies on a similar intuition as equivalence by session, adding (much stronger) structural requirements to proofs. To prove diff-equivalence of P and Q, one first requires that P and Q have *syntactically* the same structure and that they only differ by the data (i.e. the terms) inside the process. Second, any trace of P must be matched in Q by the trace that follows exactly the same control flow. Consider for example

$$P = \overline{c}\langle u\rangle \mid \overline{c}\langle v\rangle \mid R \qquad\qquad Q = \overline{c}\langle u'\rangle \mid \overline{c}\langle v'\rangle \mid R'$$

For P and Q to be diff-equivalent, traces of P starting with $\overline{c}\langle u\rangle$ need to be matched by traces of Q starting with $\overline{c}\langle u'\rangle$.

In the original definition of diff-equivalence in [BAF05] the conditional branchings were also required to result into the same control-flow. This condition has however been relaxed within [CB13]: the resulting diff-equivalence can be defined in our formalism as equivalence by session in which the rule (Match) only performs the identity matching. That is, if we write $\mathbb{T}_d(P, Q)$ for the subset of traces of $\mathbb{T}(P, Q)$ where rule (Match) is replaced by

$$\begin{aligned}&(\{\!\{(P_1 \mid \cdots \mid P_n, Q_1 \mid \cdots \mid Q_n)\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1)\\ &\qquad \xrightarrow{\tau}_s (\{\!\{(P_i, Q_i)\}\!\}_{i=1}^n \cup \mathcal{P}^2, \Phi_0, \Phi_1)\end{aligned}$$

then we define $P \sqsubseteq_d Q$ as the statement

$$\forall t \in \mathbb{T}(P),\ \exists t^2 \in \mathbb{T}_d(P, Q),\ t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)\,.$$

We say that P and Q are diff-equivalent, written $P \approx_d Q$, when $P \sqsubseteq_d Q$ and $Q \sqsubseteq_d P$. By definition $\mathbb{T}_d(P, Q) \subseteq \mathbb{T}(P, Q)$ and diff-equivalence therefore refines equivalence by session. The converse does not hold in general, e.g.

$$P = \overline{c}\langle a\rangle \mid \overline{c}\langle b\rangle \qquad Q = \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle \qquad a, b \in \mathcal{N}_{pub} \text{ distinct}$$

This example is extreme as a pre-processing on parallel operators would make the processes diff-equivalent. Such a pre-processing is however not possible for more involved, real-world examples such as the equivalences we prove on the BAC protocol in Section 7. The reason is that the matchings have to be selected dynamically, that is, different session matchings are needed to match different traces.

Relation to observational equivalence As a side result we also compare equivalence by session to observational equivalence $\approx_o$, or technically to the equivalent notion of *labelled bisimilarity* as described in [ABF18, CKR18a]. Just as equivalence by session, it is known to be an intermediate refinement between diff-equivalence and trace equivalence [CD09]:

Lemma 3.3. $\approx_d \subseteq \approx_o \subseteq \approx_{tr}$. Besides, $\approx_o$ and $\approx_{tr}$ coincide for determinate processes.

In particular by Proposition 3.2 we obtain that the trace, session, and observational equivalences coincide for determinate processes. However they are incomparable in general:

Lemma 3.4. $\approx_s$ and $\approx_o$ are incomparable.

Proof. If we write $P = c(x).c(x)$ and $Q = c(x) \mid c(x)$, then $P \approx_o Q$ but $P \not\approx_s Q$. Besides, if $k_0, k_1, k_2 \in \mathcal{N}_{priv}$ we define

$$\begin{aligned} R(t_0,t_1,t_2) = \quad & \bar{c}\langle k_0\rangle \mid \bar{c}\langle k_1\rangle \mid \bar{c}\langle k_2\rangle \mid \\ & c(x).\ \mathsf{if}\ x = k_0\ \mathsf{then}\ \bar{c}\langle t_0\rangle \\ & \qquad \mathsf{else\ if}\ x = k_1\ \mathsf{then}\ \bar{c}\langle t_1\rangle \\ & \qquad \mathsf{else\ if}\ x = k_2\ \mathsf{then}\ \bar{c}\langle t_2\rangle \end{aligned}$$

If $a, b \in \mathcal{N}_{pub}$ distinct, we have $R(a,b,b) \approx_s R(b,a,a)$ but $R(a,b,b) \not\approx_o R(b,a,a)$. □

To sum up the relations between all equivalences:

Proposition 3.5. If $\approx \in \{\approx_o, \approx_s\}$ then $\approx_d \subsetneq \approx \subsetneq \approx_{tr}$ and, for determinate processes, $\approx = \approx_{tr}$.

3.3 Trace refinements

We present an abstract notion of *optimisation*, based on trace refinements. This comes with several properties on how to compose them, providing a unified way of presenting different concrete optimisations for the decision of equivalence by session.

Definition 3.2. An *optimisation* is a pair $\mathbb{O} = (\mathbb{O}^\forall, \mathbb{O}^\exists)$ with $\mathbb{O}^\forall$ a set of traces of extended processes (*universal optimisation*), and $\mathbb{O}^\exists$ a set of traces of extended twin-processes (*existential optimisation*).

Intuitively, an optimisation reduces the set of traces that are considered when verifying equivalence: when proving $P \sqsubseteq_s Q$, only traces of $\mathbb{T}(P) \cap \mathbb{O}^\forall$ and $\mathbb{T}(P,Q) \cap \mathbb{O}^\exists$ will be studied. That is, we define the equivalence $\approx_{\mathbb{O}} = \sqsubseteq_{\mathbb{O}} \cap \sqsupseteq_{\mathbb{O}}$ where $P \sqsubseteq_{\mathbb{O}} Q$ means

$$\forall t \in \mathbb{T}(P) \cap \mathbb{O}^\forall,\ \exists t^2 \in \mathbb{T}(P,Q) \cap \mathbb{O}^\exists,\ t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)\,.$$

Note that for a few of our refinements (Section 5), the definition of $\mathbb{O}^\forall$ may depend on both P and Q.

In particular $\approx_{\mathbb{O}_{\mathsf{all}}}$ is the equivalence by session, where $\mathbb{O}_{\mathsf{all}} = (\mathbb{O}^\forall_{\mathsf{all}}, \mathbb{O}^\exists_{\mathsf{all}})$ contains all traces. Of course, such refinements may induce different notions of equivalence, hence the need for correctness arguments specific to each layer of optimisation. We specify this as follows: if $\mathbb{O}_\alpha = (\mathbb{O}^\forall_\alpha, \mathbb{O}^\exists_\alpha)$ and $\mathbb{O}_\beta = (\mathbb{O}^\forall_\beta, \mathbb{O}^\exists_\beta)$, we say that $\mathbb{O}_\alpha$ is a *correct refinement* of $\mathbb{O}_\beta$ when

$$\mathbb{O}^\forall_\alpha \subseteq \mathbb{O}^\forall_\beta \quad \text{and} \quad \mathbb{O}^\exists_\alpha \subseteq \mathbb{O}^\exists_\beta \quad \text{and} \quad \approx_{\mathbb{O}_\alpha} = \approx_{\mathbb{O}_\beta}\,.$$

Correct refinements contribute to reducing the complexity of deciding equivalence.

Properties The remainder of this section provides elementary properties useful when constructing, and composing optimisations. First we show that they can be constructed stepwise.

Proposition 3.6 (transitivity). If $\mathbb{O}_1$ is a correct refinement of $\mathbb{O}_2$, and $\mathbb{O}_2$ is a correct refinement of $\mathbb{O}_3$, then $\mathbb{O}_1$ is a correct refinement of $\mathbb{O}_3$.

Moreover, we can prove universal and existantial optimisations in a modular way:

Proposition 3.7 (combination). If $(\mathbb{O}^\forall_{opt}, \mathbb{O}^\exists)$ and $(\mathbb{O}^\forall, \mathbb{O}^\exists_{opt})$ are correct refinements of $(\mathbb{O}^\forall, \mathbb{O}^\exists)$, then $(\mathbb{O}^\forall_{opt}, \mathbb{O}^\exists_{opt})$ is a correct refinement of $(\mathbb{O}^\forall, \mathbb{O}^\exists)$.

Proof. Let $\approx_{\times\times}$, $\approx_{\circ\times}$, $\approx_{\times\circ}$ and $\approx_{\circ\circ}$ the equivalences induced by $(\mathbb{O}^\forall, \mathbb{O}^\exists)$, $(\mathbb{O}^\forall_{opt}, \mathbb{O}^\exists)$, $(\mathbb{O}^\forall, \mathbb{O}^\exists_{opt})$ and $(\mathbb{O}^\forall_{opt}, \mathbb{O}^\exists_{opt})$, respectively. As $\approx_{\circ\times} = \approx_{\times\times} = \approx_{\times\circ}$ by hypothesis, the result follows from the straightforward inclusions $\approx_{\circ\circ} \subseteq \approx_{\circ\times}$ and $\approx_{\times\circ} \subseteq \approx_{\circ\circ}$. □

Relying on this result, we see a universal optimisation $\mathbb{O}^\forall$ (resp. existential optimisations $\mathbb{O}^\exists$) as the optimisation $(\mathbb{O}^\forall, \mathbb{O}^\exists_{\mathsf{all}})$ (resp. $(\mathbb{O}^\forall_{\mathsf{all}}, \mathbb{O}^\exists)$). This lightens presentation as we can now meaningfully talk about universal (resp. existential) optimisations being correct refinements of others. Finally, when implementing such optimisations in tools, deciding the membership of a trace in the sets $\mathbb{O}^\forall$ or $\mathbb{O}^\exists$ may sometimes be inefficient or not effective. In these cases we may want to implement these optimisations partially, using for example sufficient conditions. The following proposition states that such partial implementations still result into correct refinements.

Proposition 3.8 (partial implementability). Let us consider the optimisations $\mathbb{O}^\forall_{opt} \subseteq \mathbb{O}^\forall_{part} \subseteq \mathbb{O}^\forall$ and $\mathbb{O}^\exists_{opt} \subseteq \mathbb{O}^\exists_{part} \subseteq \mathbb{O}^\exists$. If $\mathbb{O}^\forall_{opt}$ is a correct refinement of $\mathbb{O}^\forall$ and $\mathbb{O}^\exists_{opt}$ is a correct refinement of $\mathbb{O}^\exists$, then $(\mathbb{O}^\forall_{part}, \mathbb{O}^\exists_{part})$ is a correct refinement of $(\mathbb{O}^\forall, \mathbb{O}^\exists)$.

In the rest of the paper we assume the reader familiar with group theory (group actions, stabilisers), in particular the group of permutations (written in cycle notation). Most of our optimisations are indeed expressed using this terminology.

4 PARTIAL-ORDER REDUCTIONS

In this section we present partial-order reductions for equivalence by session. They are inspired by similar techniques developed for proving trace equivalence of determinate processes [BDH15], although they differ in their technical development to preserve correctness in our more general setting. In particular the optimisations we present account for non determinacy and private channels which will be useful when analysing e-voting protocols.

4.1 Labels and independence

Labels Partial-order reduction techniques identify commutativity relations in a set of traces and factor out the resulting redundancy. Here we exploit the permutability of concurrent actions without output-input data flow. For that we introduce *labels* to reason about dependencies in the execution:

- Plain processes P are labelled $[\,P\,]^{\ell}$, with ℓ a word of integers reflecting the position of P within the whole process.
- Actions α are labelled $[\,\alpha\,]^{L}$ to reflect the label(s) of the process(es) they originate from. That is, L is either a single integer word ℓ (for inputs and outputs) or a pair of such, written $\ell_1 \mid \ell_2$ (for internal communications).

Labels can be bootstrapped arbitrarily, say, by the empty word ε, and are propagated as follows in the operational semantics. The (Par) rule extends labels:

$$(\{\!\{[\,P_1 \mid \cdots \mid P_n\,]^{\ell}\}\!\} \uplus \mathcal{P}, \Phi) \xrightarrow{[\,\tau\,]^{\ell}}_s (\{\!\{[\,P_i\,]^{\ell.i}\}\!\}_{i=1}^{n} \uplus \mathcal{P}, \Phi)$$

the rules (In) and (Out) preserve labels:

$$(\{\!\{[\,P\,]^{\ell}\}\!\} \uplus \mathcal{P}, \Phi) \xrightarrow{[\,\alpha\,]^{\ell}}_s (\{\!\{[\,P'\,]^{\ell}\}\!\} \uplus \mathcal{P}, \Phi')$$

and so does (Comm), however producing a double label:

$$(\{\!\{[\,P\,]^{\ell}, [\,Q\,]^{\ell'}\}\!\} \uplus \mathcal{P}, \Phi) \xrightarrow{[\,\tau\,]^{\ell|\ell'}} (\{\!\{[\,P'\,]^{\ell}, [\,Q'\,]^{\ell'}\}\!\} \uplus \mathcal{P}, \Phi)\,.$$

In particular, we always implicitly assume the invariant preserved by transitions that extended processes contain labels that are pairwise incomparable w.r.t. the prefix ordering.

Independence Labels materialise flow dependencies. Two actions $\alpha = [\,a\,]^{L}$ and $\alpha' = [\,a'\,]^{L'}$ are said *sequentially dependent* if one of the (one or two) words constituting L, and one of those constituting L', are comparable w.r.t. the prefix ordering. Regarding input-output dependencies, we say that α and α' are *data dependent* when $\{a, a'\} = \{\overline{c}\langle \mathsf{ax}\rangle, c(\xi)\}$ with ax appearing in ξ.

Definition 4.1 (independence). Two actions α and α' are said *independent*, written $\alpha \parallel \alpha'$, when they are sequentially independent and data independent.

There is some redundancy in the trace space in that, intuitively, swapping adjacent, independent actions in a trace has no substantial effect. Still, this is rather weak: for example the recipe $\mathsf{proj}_1(\langle n, \mathsf{ax}\rangle)$ is artificially dependent in the axiom ax, preventing optimisations. Such spurious dependencies can be erased using the following notion:

Definition 4.2 (recipe equivalence). Two input transitions

$$(\mathcal{P}, \Phi) \xrightarrow{[\,c(\xi_1)\,]^{\ell}} A \qquad\qquad (\mathcal{P}, \Phi) \xrightarrow{[\,c(\xi_2)\,]^{\ell}} A$$

are said *recipe equivalent* when $\xi_1\Phi =_E \xi_2\Phi$. Two traces are recipe equivalent if one can be obtained from the other by replacing some transitions by recipe-equivalent ones.

The rest of this section formalises the intuition that equivalence by session can be studied up to recipe-equivalent rewriting of traces, and arbitrary permutation of their independent actions. Proofs can be found in Appendix C.1.

Correctness of por techniques If $\mathsf{tr} = \alpha_1 \cdots \alpha_n$ and π is a permutation of $[\![1, n]\!]$, we write

$$\pi.\mathsf{tr} = \alpha_{\pi(1)} \cdots \alpha_{\pi(n)}\,.$$

This is an action of the group of permutations of $[\![1, n]\!]$ on action words of size n. We say that π *permutes independent actions* of tr if either $\pi = \mathsf{id}$, or $\pi = \pi_0 \circ (i \;\; i+1)$ with $\alpha_i \parallel \alpha_{i+1}$ and π_0 permutes independent actions of $(i \;\; i+1).\mathsf{tr}$. Such permutations preserve the group structure of permutations, in the sense of these two straightforward propositions:

Proposition 4.1 (composition). If π permutes independent actions of tr, and π' permutes independent actions of $\pi.\mathsf{tr}$, then $\pi' \circ \pi$ permutes independent actions of tr.

Proposition 4.2 (inversion). If π permutes independent actions of tr, then π^{-1} permutes independent actions of $\pi.\mathsf{tr}$.

We will use these two properties implicitly in many proofs. But more importantly, the action of permutations on trace words can be lifted to traces:

Proposition 4.3. If $t : A \overset{\mathsf{tr}}{\Longrightarrow} B$ and π permutes independent actions of tr, then $A \overset{\pi.\mathsf{tr}}{\Longrightarrow} B$. This trace is unique if we take labels into account, and will be referred as $\pi.t$.

Together with recipe equivalence, this is the core notion for defining partial-order reductions. We gather them into $\equiv_{\mathsf{por}}$ the smallest equivalence relation over traces containing recipe equivalence and such that $t \equiv_{\mathsf{por}} \pi.t$ when π permutes independent actions of t. The result below justifies that quotients by $\equiv_{\mathsf{por}}$ result in correct refinements.

Proposition 4.4 (correctness of por). Let $\mathbb{O}_1^{\forall} \subseteq \mathbb{O}_2^{\forall}$ be universal optimisations. We assume that for all $t \in \mathbb{O}_2^{\forall}$, there exists t_{ext} such that t is a prefix of t_{ext}, and $t' \in \mathbb{O}_1^{\forall}$ such that $t' \equiv_{\mathsf{por}} t_{ext}$. Then $\mathbb{O}_1^{\forall}$ is a correct refinement of $\mathbb{O}_2^{\forall}$.

4.2 Compression optimisations

We first present a compression of traces into blocks of actions of a same type (inputs, outputs and parallel, or internal communications) by exploiting Proposition 4.4. We formalise this idea by using reduction strategies based on *polarity* patterns.

Polarities and phases We assign polarities to processes depending on their toplevel actions: public inputs are positive (+1), public outputs and parallels are overwhelmingly negative ($-\infty$), and others are null.

$$\begin{array}{lll} polar(c(x).P) = 1 & polar(\overline{c}\langle u\rangle.P) = -\infty & c \in Ch_{pub} \\ polar(d(x).P) = 0 & polar(\overline{d}\langle u\rangle.P) = 0 & d \in Ch_{priv} \\ polar(0) = 0 & polar(P \mid Q) = -\infty & \end{array}$$

This notion is lifted to extended processes by summing:

$$polar((\mathcal{P}, \Phi)) = \sum_{R \in \mathcal{P}} polar(R)\,.$$

In particular, extended processes containing an executable parallel operator or output has polarity $-\infty$, and executing public inputs makes polarity nonincreasing. We then identify the trace patterns at the core of our partial-order reductions. We say that a trace

$$t : A_0 \xrightarrow{[\,a_1\,]^{L_1}} \cdots \xrightarrow{[\,a_n\,]^{L_n}} A_n$$

- is a *negative phase* when all transitions are outputs or parallels, and $polar(A_n) \neq -\infty$.
- is a *null phase* when $polar(A_0) \geqslant 0$, $n = 1$ and the transition is an internal communication.
- is a *positive phase* when $polar(A_0) > 0$, all transitions are inputs, all L_i's are equal, and $polar(A_0) > polar(A_n)$.

Rephrasing, a negative phase executes all available outputs and parallels, a null phase is one internal communication, and a positive phase executes a whole chain of inputs. Note that only negative phases may be empty.

Basic compression The first optimisation is to only consider traces that can be decomposed into phases. Formally we write $\mathbb{O}^{\forall}_{\mathsf{c},b}$ the set of traces of the form

$$t : b_0^- \cdot b_1^+ \cdot b_1^- \cdot b_2^+ \cdot b_2^- \cdots b_n^+ \cdot b_n^-$$

where each b_i^+ is a positive or null phase, and each b_i^- is a negative phase. We show in Appendix C.3 that any maximal trace can be decomposed this way after application of a well-chosen permutation of independent actions. Hence by Proposition 4.4:

Proposition 4.5 . $\mathbb{O}^{\forall}_{\mathsf{c},b}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{all}}$.

Determinism of negative phases Negative phases are nondeterministic by essence, but the underlying combinatorial explosion is artificial in that most of the actions within negative phases are independent: we show that they can actually be executed purely deterministically.

We fix an arbitrary total ordering $\preccurlyeq$ on labelled actions. A negative phase b^-, with $tr(b^-) = \alpha_1 \cdots \alpha_n$, is said *consistent* when for all $i < n$ such that $\alpha_i \parallel \alpha_{i+1}$, we have $\alpha_i \preccurlyeq \alpha_{i+1}$. We write $\mathbb{O}^{\forall}_{\mathsf{c}}$ the subset of $\mathbb{O}^{\forall}_{\mathsf{c},b}$ of traces whose negative phases are all consistent.

Proposition 4.6 . $\mathbb{O}^{\forall}_{\mathsf{c}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c},b}$.

Proof. By Proposition 4.4, it suffices to prove that for all negative phases b^-, there exists π permuting independent actions of b^- such that $\pi.b^-$ is consistent. This follows from a well-founded induction on $tr(b^-)$ w.r.t the lexicographic extension of $\preccurlyeq$ on words of actions. □

Blocks We have established that, to prove equivalences by session, it is sufficient to consider traces of a certain shape, namely, traces alternating between nonnegative and negative phases. The rest of our partial-order reductions reason at the granularity of these phases, by reordering so-called *blocks*.

A block is a positive or null phase followed by a negative phase. Any trace of $\mathbb{O}^{\forall}_{\mathsf{c}}$ is therefore composed of an initial negative phase and a sequence of blocks. Two blocks b and b' are said *independent*, written $b \parallel b'$, if all actions of the former are independent of all actions of the latter. Analogously to actions, we refer to permutations π permuting independent blocks of traces of $\mathbb{O}^{\forall}_{\mathsf{c}}$. All related notations and results can be cast to blocks by using:

Proposition 4.7 . Let $t : b_p \cdots b_n$ a sequence of blocks, $\mathsf{tr}_i = tr(b_i)$. If π permutes independent blocks of $\mathsf{tr} = tr(t)$, then there is π' permuting independent actions of tr s.t.

$$\pi'.\mathsf{tr} = \pi.\mathsf{tr} = \mathsf{tr}_{\pi(p)} \cdots \mathsf{tr}_{\pi(n)}\,.$$

Note in particular the following corollary of Proposition 4.4 that will be at the core of the results of the next sections, where $\equiv_{\text{b-por}}$ is the analogue of $\equiv_{\text{por}}$ where permutation of independent actions is replaced by permutation of independent blocks:

Corollary 4.8 . Let $\mathbb{O}^{\forall}_1 \subseteq \mathbb{O}^{\forall}_2 \subseteq \mathbb{O}^{\forall}_{\mathsf{c}}$. We assume that for all $t \in \mathbb{O}^{\forall}_2$, there exists $t_{ext} \in \mathbb{O}^{\forall}_{\mathsf{c}}$ such that t is a prefix of t_{ext}, and $t' \in \mathbb{O}^{\forall}_1$ such that $t' \equiv_{\text{b-por}} t_{ext}$. Then $\mathbb{O}^{\forall}_1$ is a correct refinement of $\mathbb{O}^{\forall}_2$.

4.3 Improper positive phases

We now introduce *improper blocks*: intuitively, they conclude the execution of a process but do not bring new knowledge to the attacker. Such blocks can always be relegated to the end of traces because they are not essential to execute other blocks. Formally, we say that a block

$$b : (\mathcal{P}, \Phi) \overset{\mathsf{tr}}{\Longrightarrow} (Q, \Phi \cup \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\})$$

is *improper* if

(1) it starts with an input, i.e. $\mathsf{tr} = c(\xi) \cdot \mathsf{tr}'$ for some ξ, tr'; and
(2) all labels appearing in tr do not appear in Q, except maybe on null processes; and
(3) for all $i \in [\![1, n]\!]$, t_i is deducible from Φ, that is, there exists a recipe ξ_i such that $\xi_i\Phi =_E t_i$.

This takes inspiration, but generalises, the notion of improper blocks used in [BDH15, CKR19] that requires $n = 0$. Our finer optimisation captures for example outputs of public error codes in the model of the e-passport in Example 2.2 (error, ok). Let us also highlight the first item of the definition discarding improper blocks starting with an internal communication: although the correctness statement below would still hold without this restriction, this is not the case when combined with the optimisations of the next sections. They advance the execution of some blocks starting with an internal communication: we shall therefore ensure that such blocks are never deemed improper since, as such, they would also have to be delayed.

Formally we write $\mathbb{O}^{\forall}_{\mathsf{c+i}}$ the subset of $\mathbb{O}^{\forall}_{\mathsf{c}}$ of traces not containing an improper block followed by a proper block. The following results, proved in Appendix C.4, justify its correctness.

Proposition 4.9 . For all $t \in \mathbb{O}^{\forall}_{\mathsf{c}}$, there exists $t' \equiv_{\text{b-por}} t$ such that $t' \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ and t' has the same number of improper blocks t.

The fact that t' has the same number of improper blocks as t is not necessary to apply Corollary 4.8, but it will help establishing further optimisations on improper blocks in the next section.

Corollary 4.10 . $\mathbb{O}^{\forall}_{\mathsf{c+i}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c}}$.

Note that when restricting the definition to $n = 0$, we obtain a weaker optimisation than the one presented in [BDH15] for determinate processes. The latter indeed considers traces with at most one improper block. This, however, relies on determinate-specific arguments that are unsound for equivalence by session in general.

4.4 Stabilising proper blocks

We introduce in this section a refinement $\mathbb{O}^{\forall}_{\mathsf{c+i^*}} \subseteq \mathbb{O}^{\forall}_{\mathsf{c+i}}$ of improper blocks. It is not included in our prototype as we found it quite hard to implement; it should rather be seen as a convenient proof tool in that, for incoming optimisations $\mathbb{O}^{\forall}$, it will be easier to prove the correctness of $\mathbb{O}^{\forall} \cap \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ rather than $\mathbb{O}^{\forall}$ directly.

Intuitively, it solves the problem that properness is not stable by permutation of independent blocks. For example consider processes built from signatures and names

$$\mathcal{F} = \{\mathsf{h}/1, a/0\} \qquad c \in Ch_{pub} \qquad k \in \mathcal{N}_{priv}$$

modelling a hash function h and a constant a, and the process

$$P = c(x).\bar{c}\langle \mathsf{h}(k)\rangle.0 \mid c(x).\bar{c}\langle k\rangle.0$$

The traces of P can contain two blocks, one outputting $\mathsf{h}(k)$ and one outputting k: the former may be improper or not depending on whether it is executed second or first. This highlights that permutations of independent, *proper* blocks of a trace $t \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ may lead to a trace $\pi.t \notin \mathbb{O}^{\forall}_{\mathsf{c+i}}$. To get around this issue we let $\mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ the set of traces of the form $u \cdot v$ where $u, v \in \mathbb{O}^{\forall}_{\mathsf{c}}$ and

- for all π permuting independent blocks of u, $\pi.u$ does not contain any improper blocks;
- v only contains improper blocks.

In particular this optimisation has the important property of being stable under permutation of proper blocks:

Proposition 4.11. Let $t \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ with $t = u \cdot v$ and u containing only proper blocks and v, only improper blocks. Then for all π permuting independent blocks of u, $(\pi.u \cdot v) \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$.

We then prove the correctness of this optimisation.

Proposition 4.12. For all $t \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$, there exists $t' \equiv_{\text{b-por}} t$ such that $t' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$.

Proof. We proceed by induction on the number of proper blocks of t. If $t \in \mathbb{O}^{\forall}_{1}$ (which subsumes the case where t has no proper blocks) it sufficies to choose $t' = t$. Otherwise let us write $t = u \cdot v$ with u containing only proper blocks and v improper blocks, and let π permuting independent blocks of u such that $\pi.u$ contains an improper block. By Proposition 4.9 there exists $u' \equiv_{\text{b-por}} \pi.u$ such that $u' \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ and u' contains the same number of improper blocks as $\pi.u$. In particular if we write $s = u' \cdot v$, we have $s \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$, s has less proper blocks than t, and $s \equiv_{\text{b-por}} t$. Hence the conclusion by induction hypothesis applied to s. □

Corollary 4.13. $\mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c+i}}$.

4.5 High-priority null phases

In this section we introduce a condition for prioritising the execution of some internal communications. Consider for example

$$c(x).P_1 \mid \bar{d}\langle u\rangle.P_2 \mid \bar{d}\langle v\rangle.P_3 \mid d(x).P_4$$

where $c \in Ch_{pub}$ and $d \in Ch_{priv}$. Assuming that the channel d does not appear in P_1, all (maximal) traces will contain an internal communication on d between the rightmost three processes, regardless of the potential prior execution of $c(x).P_1$. Our optimisation will typically execute such internal communications in priority. Formally let A be an extended process with $polar(A) \geqslant 0$

$$A = (\{\!\!\{ [\, P_1 \,]^{\ell_1}, \ldots, [\, P_n \,]^{\ell_n} \}\!\!\}, \Phi)\,.$$

If $d \in Ch_{priv}$ is a private channel, we write $\mathbb{L}_A(d) \subseteq \{\ell_1, \ldots, \ell_n\}$ the set of labels ℓ_i such that P_i starts with an input or output on d. When an internal communication is possible on $d \in Ch_{priv}$, we say that d is *high-priority* (in A) if for all traces of the form

$$A \stackrel{\mathsf{tr}}{\Longrightarrow} B \qquad \text{where no labels of } \mathbb{L}_A(d) \text{ appear in tr}$$

we have $\mathbb{L}_A(d) = \mathbb{L}_B(d)$. This formalises that, in all traces of A, the first internal communication on d needs be between an input and an output that were already available in A.

By extension, a transition is high-priority when it is an internal communication on the minimal high-priority channel w.r.t. an arbitrary total ordering $\preccurlyeq_{Ch}$ on Ch_{priv}. Rephrasing, it is a null phase on one, deterministically-chosen, high-priority channel. We then define $\mathbb{O}^{\forall}_{0}$ the subset of traces of $\mathbb{O}^{\forall}_{\mathsf{c}}$ that do no contain non-high-priority transitions from processes from which a high-priority transition is possible. To combine it with the previous optimisations we let $\mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ the set of traces of the form

$$b^- \cdot t_p \cdot t_i \;\in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$$

where b^- is a negative phase, $t_p \in \mathbb{O}^{\forall}_{0}$ only contains proper blocks, and $t_i \in \mathbb{O}^{\forall}_{\mathsf{c}}$ only contains improper blocks. The following propositions state the correctness of this optimisation; their proofs are detailed in Appendix C.5 and rely as usual on Corollary 4.8.

Proposition 4.14. Let $t = u \cdot v$ a maximal trace, where $u, v \in \mathbb{O}^{\forall}_{\mathsf{c}}$ and v does not contain any high-priority transitions. Then there exists π permuting independent blocks of u such that $\pi.u \in \mathbb{O}^{\forall}_{0}$.

Corollary 4.15. $\mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c+i^*}}$.

4.6 Reduction of independent blocks

Finally, as sequences of independent blocks can be permuted arbitrarily, we define an optimisation that fixes their order. For that we let $\preccurlyeq$ an ordering on words of actions such that two words of independent actions are always strictly comparable ($\prec$). We then define a predicate $\mathsf{Minimal}(t, b)$, ensuring that it is not possible to obtain an action word lexicographically-smaller than $tr(t \cdot b)$ by inserting b inside t using permutations of independent blocks.

$$\begin{array}{lll} \mathsf{Minimal}(b_1 \cdots b_n, b) & & \text{if } n = 0 \\ & & \text{or } \neg(b_n \parallel b) \\ & & \text{or } b \textit{ can follow } b_n \text{ (see below)} \\ & & \quad \text{and } \mathsf{Minimal}(b_1 \cdots b_{n-1}, b) \end{array}$$

Intuitively, we say that a block b can follow an other block b_n when $tr(b_n) \prec tr(b)$, but this test may be ignored in some cases to ensure compatibility prior optimisations. Formally, b can follow b_n when

(1) $tr(b_n) \prec tr(b)$; or
(2) b_n starts with a high-priority transition but not b; or
(3) both b_n and b start with a high-priority transition, but on different private channels.

We define $\mathbb{O}^{\forall}_{\mathrm{r}}$ the smallest subset of $\mathbb{O}^{\forall}_{\mathrm{c}}$ containing the empty trace, and such that $t \in \mathbb{O}^{\forall}_{\mathrm{r}}$ and $\mathsf{Minimal}(t, b)$ implies $t \cdot b \in \mathbb{O}^{\forall}_{\mathrm{r}}$. To account for improper blocks, we let $\mathbb{O}^{\forall}_{\mathrm{por}}$ the set of traces

$$b^- \cdot t_p \cdot t_i \ \in \mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$$

where b^- is a negative phase, $t_p \in \mathbb{O}^{\forall}_{\mathrm{r}}$ only contains proper blocks, and $t_i \in \mathbb{O}^{\forall}_{\mathrm{r}}$ only contains improper blocks. The correctness of this optimisation is proved in Appendix C.6.

Proposition 4.16 . For all maximal traces $t \in \mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$, there exists π permuting independent blocks of t such that $\pi.t \in \mathbb{O}^{\forall}_{\mathrm{por}}$. Besides $\pi.t \prec_{lex} t$ if $t \notin \mathbb{O}^{\forall}_{\mathrm{por}}$, with $\preccurlyeq_{lex}$ the lexicographic extension of $\preccurlyeq$.

The decreasing argument w.r.t. $\preccurlyeq_{lex}$ is not necessary to obtain correctness, but will be important to prove the compatibility with the symmetry-based optimisations presented in the next section.

Corollary 4.17 . $\mathbb{O}^{\forall}_{\mathrm{por}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$.

5 REDUCTIONS BY SYMMETRY

In this section, we show how to exploit process symmetries for equivalence by session. Such symmetries often appear in practice when we verify multiple sessions of a same protocol as it results into parallel copies of identical processes, up to renaming of fresh names. We first provide a group-theoretical characterisation of internal process redundancy, and then design two optimisations. In spirit, this approach shares some similarities with classical work in model checking modelling symmetries by the group of the automorphisms of the system to be analysed [ES96].

5.1 Structural equivalence

We exhibit an equivalence identifying processes that have an identical structure (up to associativity and commutativity of parallel operators) and whose data are equivalent w.r.t. the equational theory and alpha-renaming of private names. This will be the basis of our symmetry-based refinements. We define *structural equivalence* $\equiv_{\mathrm{ac}}$ on plain processes as the smallest equivalence such that

$$P \mid Q \equiv_{\mathrm{ac}} Q \mid P \qquad\qquad (P \mid Q) \mid R \equiv_{\mathrm{ac}} P \mid (Q \mid R)$$

and that is closed under context (that is, composition of equivalent processes with either a same process in parallel, or an input, output, or conditional instruction at toplevel). To account for the equational theory, we extend it to $\equiv_E$ defined by

$$\frac{\sigma, \sigma' \text{ substitutions} \qquad P \equiv_{\mathrm{ac}} Q \qquad \forall x \in \mathcal{X}, x\sigma =_E x\sigma'}{P\sigma \equiv_E Q\sigma'}$$

Besides we add alpha equivalence of private names and channels; intuitively, two agents executing the same protocol are behaving similarly even though they use their own session nonces. Formally we define *structural equivalence* $\equiv$ on extended processes by the following inference rule

$$\frac{\forall i, P_i \equiv_E Q_i \qquad \varrho \ \ \alpha\text{-renaming}}{(\{\!\!\{ [\,P_1\,]^{\ell_1}, \ldots, [\,P_n\,]^{\ell_n} \}\!\!\}, \Phi) \equiv (\{\!\!\{ [\,Q_1\,]^{\ell_1}, \ldots, [\,Q_n\,]^{\ell_n} \}\!\!\}, \Phi)\,\varrho}$$

where an *α-renaming* is a substitution for channels and names $\varrho = \varrho_{\mathcal{N}} \circ \varrho_{Ch}$ for some permutations $\varrho_{\mathcal{N}}$ and ϱ_{Ch} of $\mathcal{N}_{priv}$ and Ch_{priv}.

5.2 Group actions and process redundancy

Let a labelled extended process (as defined in Section 4.1) and a labelled extended twin process

$$A = (\{\!\!\{ [\,P_1\,]^{\ell_1}, \ldots, [\,P_n\,]^{\ell_n} \}\!\!\}, \Phi) \qquad \ell_1 < \cdots < \ell_n$$
$$A^2 = (\{\!\!\{ ([\,P_1\,]^{\ell_1}, Q_1), \ldots, ([\,P_n\,]^{\ell_n}, Q_n) \}\!\!\}, \Phi_0, \Phi_1)$$

where $\preccurlyeq$ is an arbitrary total ordering on labels. We also let $\pi \in S_n$ where S_n denotes the the group of all permutations of $[\![1, n]\!]$. We define the following group action of S_n on extended (twin) process:

$$\pi.A = (\{\!\!\{ [\,P_{\pi(1)}\,]^{\ell_1}, \ldots, [\,P_{\pi(n)}\,]^{\ell_n} \}\!\!\}, \Phi)$$
$$\pi.A^2 = (\{\!\!\{ ([\,P_{\pi(1)}\,]^{\ell_1}, Q_1), \ldots, ([\,P_{\pi(n)}\,]^{\ell_n}, Q_n) \}\!\!\}, \Phi_0, \Phi_1)$$

In the case of twin processes, this notation is only well defined if for all $i \in [\![1, n]\!]$, $\mathsf{skel}(P_{\pi(i)}) = \mathsf{skel}(Q_i)$, i.e. if $\mathsf{skel}(Q_i) = \mathsf{skel}(Q_{\pi(i)})$. The labels are only used as a way to fix the ordering of the processes within the multiset. They will often be omitted for succinctness, assuming an implicit ordering.

Process redundancy within an extended process is then simply captured by the group stabiliser

$$\mathsf{Stab}(A) = \{\pi \in S_n \mid \pi.A \equiv A\}\,.$$

Example 5.1. $\mathsf{Stab}(\{\!\!\{P, \ldots, P\}\!\!\}, \Phi) = S_n$ models the case where all parallel subprocesses are identical. On the contrary, the case where $\mathsf{Stab}(A) = \{\mathsf{id}\}$ models that there is no redundancy at all between parallel processes. Intermediate examples model partial symmetries: the larger the stabiliser, the more redundancy. For example $\mathsf{Stab}(\{\!\!\{P, P, Q, Q, Q\}\!\!\}, \Phi)$ will contain at least the subgroup of S_n generated by the permutations $(1\ \ 2)$, $(3\ \ 4)$ and $(3\ \ 5)$. △

If ϱ is a permutation of Ch_{pub}, we also generalise the definition of $\mathsf{Stab}(A)$ as follows:

$$\mathsf{Stab}_{\varrho}(A) = \{\pi \in S_n \mid \pi.A \equiv A\varrho\}\,.$$

This characterises symmetries up to renaming of public channels. Indeed such channels only affect the skeletons and the labels of the trace, not the execution flow: therefore two processes that are structurally-equivalent but differ on their public channels still have a similar set of traces. Typically to prove the equivalence of $P(c) \mid P(d)$ and $Q(c) \mid Q(d)$ where $P(x), Q(x)$ are processes operating on a single channel x, matching a trace starting with an action of either $P(c)$ or $P(d)$ results into a similar analysis, provided $c, d \in Ch_{pub}$.

5.3 Universal symmetry optimisations

We first present a universal optimisation that reduces the number of possibilities when applying rules (In) and (Comm) at the start of a nonnegative phase. It captures the idea that, when considering the traces of several parallel protocol sessions, starting the trace by an action from one session or an other does not make a substantial difference, even when they use distinct public channels. To formalise this idea, let us consider a labelled trace $t \in \mathbb{O}^{\forall}_{\mathrm{c}}$:

$$t : [\,P\,]^{\varepsilon} \overset{\mathsf{tr}}{\Longrightarrow} (\{\!\!\{ [\,P_i\,]^{\ell_i} \}\!\!\}_{i=1}^{n}, \Phi_0) = A\,.$$

The goal is to exhibit conditions discarding some of the possible transitions directly following t.

5.3.1 Characterisation using stabilisers

We define two sets Sym_1 and Sym_2 characterising symmetries (and prove in Appendix D.2 that they are permutation groups). They model that the symmetries in P shall be reflected in one way or an other in $\mathbb{T}(P, Q)$ to be exploited.

Symmetry by matching We first define a notion of symmetry within P that is reflected in the matching with Q. The set of symmetries we consider is the group $\mathsf{Sym}_1 = \mathsf{Sym}_1^P \cap \mathsf{Sym}_1^Q$ where

$$\mathsf{Sym}_1^P = \mathsf{Stab}(A)$$
$$\mathsf{Sym}_1^Q = \bigcap_{\substack{t^2:(P,Q)\overset{\text{tr}}{\Longrightarrow} A^2 \\ t=\mathsf{fst}(t^2)\sim\mathsf{snd}(t^2)}} \{\pi \in S_n \mid (P, Q) \overset{\text{tr}}{\Longrightarrow} \pi.A^2\}$$

In the definition of Sym_1^Q, the labels are taken into account in tr (i.e. the trace leading to $\pi^{-1}.A^2$ should have t as a first projection). Intuitively $\pi \in \mathsf{Sym}_1^P$ means that for all a, P_a and $P_{\pi(a)}$ have the same traces, and $\pi \in \mathsf{Sym}_1^Q$ means that they can be matched by the same sessions of Q. In particular if $\pi \in \mathsf{Sym}_1$, executing first an action from P_a or $P_{\pi(a)}$ results into symmetric equivalence proofs.

Simultaneous symmetry We now define a notion of symmetry capturing redundancy occurring at the same time in P and Q, up to bijective renaming of public channels. For ϱ permutations of Ch_{pub}, we define $\mathsf{Sym}_2 = \bigcup_\varrho \mathsf{Sym}_2^P(\varrho) \cap \mathsf{Sym}_2^Q(\varrho)$ where

$$\mathsf{Sym}_2^P(\varrho) = \mathsf{Stab}_\varrho(A)$$
$$\mathsf{Sym}_2^Q(\varrho) = \bigcap_{\substack{t^2:(P,Q)\overset{\text{tr}}{\Longrightarrow} A^2 \\ t=\mathsf{fst}(t^2)\sim\mathsf{snd}(t^2)}} \mathsf{Stab}_\varrho(\mathsf{snd}(A^2))$$

Intuitively $\pi \in \mathsf{Sym}_2^P$ means that for all a, P_a and $P_{\pi(a)}$ have the same traces, and $\pi \in \mathsf{Sym}_2^Q$ means that all matching traces of Q have this symmetry as well. Hence, like Sym_1, $\pi \in \mathsf{Sym}_2$ captures a process symmetry reflected in the equivalence proof.

5.3.2 Definition of the optimisation

In the following, Sym is the group generated by Sym_1 and Sym_2.

For input transitions We write $I \subseteq [\![1, n]\!]$ the set of indexes i such that an input transition is applicable from P_i in $\mathbb{O}^\forall_{\mathsf{c+i^*+0}}$. If $i \in I$ we consider the orbit of i w.r.t. the action of the group Sym:

$$\mathsf{Orb}(i) = \{\pi(i) \mid \pi \in \mathsf{Sym}\} \cap I\,.$$

Intuitively starting a trace of A by an action of P_i or P_j results into a similar analysis if i and j are on the same orbit. Thus a correct optimisation is to consider only one index per orbit when listing the possible transitions from A. Yet this representant should be chosen carefully for compatibility with prior optimisations, namely $\mathbb{O}^\forall_{\mathsf{por}}$ that discards traces that are not minimal w.r.t. an ordering $\preccurlyeq$ on blocks. We should pick a representant i such that the execution of P_i induces a minimal block w.r.t. $\preccurlyeq$. For that we assume that the ordering $\preccurlyeq$ is entirely determined by the label of the first action of the block. Hence the following extension of $\preccurlyeq$ to I is well-defined:

$$i \preccurlyeq j \iff tr(b_i) \preccurlyeq tr(b_j)$$

where b_i (resp. b_j) is an arbitrary block starting with an input from P_i (resp. P_j). An input transition on process P_a following the trace t is said *well-formed* if a is minimal w.r.t. $\preccurlyeq$ within $\mathsf{Orb}(a)$.

For internal communications The optimisation is the same as the one above, except that the symmetries should apply to the input and output processes at the same time. We write $IO \subseteq [\![1, n]\!]^2$ the set of pairs of indexes (i, j) such that an internal communication is possible between (P_i, P_j) in $\mathbb{O}^\forall_{\mathsf{c+i^*+0}}$, where the input is at the start of P_i. If $(i, j) \in IO$ we consider

$$\mathsf{Orb}(i, j) = \{(\pi(i), \pi(j)) \mid \pi \in \mathsf{Sym}\} \cap IO\,.$$

Intuitively starting a trace of A by an internal communication between processes P_i, P_j or P_k, P_l results into a similar analysis if (i, j) and (k, l) are on the same orbit. Then similarly to inputs we extend the ordering $\preccurlyeq$ on blocks to IO by writing

$$(i, j) \preccurlyeq (k, l) \iff tr(b) \preccurlyeq tr(b')$$

where b (resp. b') is the block starting with the internal communication between P_i and P_j (resp. P_k and P_l). We thus qualify an internal communication between P_a and P_b following the trace t as *well-formed* if (a, b) is minimal w.r.t. $\preccurlyeq$ within $\mathsf{Orb}(a, b)$.

Correctness We say that a trace is well-formed when all its transitions (In) and (Comm) at the start of nonnegative phases are well-formed. We then define the optimisation $\mathbb{O}^\forall_{\mathsf{sym}}$ as the set of well-formed traces of $\mathbb{O}^\forall_{\mathsf{c+i^*+0}}$. Its correctness is proved in Appendix D.2.

Proposition 5.1. $\mathbb{O}^\forall_{\mathsf{por}} \cap \mathbb{O}^\forall_{\mathsf{sym}}$ is a correct refinement of $\mathbb{O}^\forall_{\mathsf{por}}$.

5.4 Existential symmetry optimisation

The goal of this optimisation is to exploit symmetries when applying the matching rule: when several processes are structurally equivalent then we do not need to consider redundant matchings. For instance, suppose that we need to match $P_1 \mid P_2$ with $Q \mid Q$. Just considering the identity permutation would be sufficient, and the permutation $(1\ \ 2)$ should be considered as redundant. Formally, let us consider an instance of the rule (Match)

$$(\{\!\{(P, Q)\}\!\} \cup \mathcal{P}^2, \Phi_0, \Phi_1) \xrightarrow{\tau}_s (\{\!\{(P_i, Q_{\pi(i)})\}\!\}_{i=1}^n \cup \mathcal{P}^2, \Phi_0, \Phi_1) = \pi^{-1}.A^2$$

with $P = P_1 \mid \cdots \mid P_n$, $Q = Q_1 \mid \cdots \mid Q_n$ and $\pi \in S_p$ where $p = n + |\mathcal{P}^2|$. We only consider S_p instead of the usual S_n for convenience as it will make the formalisation of the optimisation lighter, in particular the definition of the following relation on S_p:

$$\pi \sim \pi' \quad \textit{iff} \quad \exists u \in \mathsf{Stab}(\mathsf{snd}(A^2)),\ \pi' = \pi \circ u\,.$$

Proposition 5.2. $\sim$ is an equivalence relation on S_p.

Proof. It sufficies to prove that $\mathsf{Stab}(\mathsf{snd}(A^2))$ is a group (Reflexivity: a group of permutations contains the identity; symmetry: a group is closed by inverse; transitivity: a group is closed by composition). Consider the function $(\pi, A) \mapsto \pi.A$. It is a group action of S_p on the set of extended processes quotiented by $\equiv$. Such an action is well-defined since for all A, B containing p processes such that $A \equiv B$ and all $\pi \in S_p$, we have $\pi.A \equiv \pi.B$. The set $\mathsf{Stab}(A)$ is a stabiliser of this action, hence the conclusion. □

We say that an instance of rule (Match) is *well-formed* when the underlying permutation π is minimal within its equivalence class for $\sim$, w.r.t. an arbitrary total ordering on permutations. We denote by $\mathbb{O}^{\exists}_{\mathsf{sym}}$ the set of traces of extended twin-processes whose instances of rule (Match) are all well-formed. The correctness of this optimisation is stated below and proved in Appendix D.3.

Proposition 5.3. $\mathbb{O}^{\exists}_{\mathsf{sym}}$ is a correct refinement of $\mathbb{O}^{\exists}_{\mathsf{all}}$.

6 SYMBOLIC SETTING

Even though we do not consider unbounded replication, the semantics of our process calculus defines an infinite transition system due to the unbounded number of possible inputs that can be provided by the adversary. To perform exhaustive verification of such infinite systems, it is common to resort to *symbolic techniques* abstracting inputs by symbolic variables and constraints. We briefly describe in this section how our optimisations are integrated in the symbolic procedure underlying the DeepSec tool.

6.1 DeepSec's baseline procedure

Symbolic setting In the DeepSec tool [CKR18b] and its underlying theory [CKR18a], the deduction capabilities of the attacker are represented by so-called *deduction facts* $X \vdash^? u$, intuitively meaning that the attacker is able to deduce the term u by the means of a recipe represented by the variable X. Additionally, conditional branching, *e.g.* if $u = v$ then ... else ..., is represented by *equations* $u =^? v$ and *disequations* $u \neq^? v$.

To represent infinitely many processes, [CKR18a] relies on *symbolic processes* $(\mathcal{P}, \Phi, C)$ where $\mathcal{P}$ and Φ are, as in our setting, a multiset of processes and a frame respectively. The difference is that the processes and frame may contain free variables: they model the variables bound by inputs and are subject to constraints in C. These constraints are a conjunction of deduction facts, equations and disequations. For example, if we consider the process

$$P = c(x).\ \mathsf{if}\ \mathsf{proj}_1(x) = t\ \mathsf{then}\ \overline{c}\langle \mathsf{h}(x)\rangle$$

then after executing symbolically the input and the positive branch of the test, we reach the symbolic process

$$(\{\!\{0\}\!\},\ \{\mathsf{ax} \mapsto \mathsf{h}(x)\},\ X \vdash^? x \wedge \mathsf{proj}_1(x) =^? t)$$

A concrete extended process is thus represented by any ground instantiation of the free variables of the symbolic process that satisfies the constraints in C. Such instantiations are called *solutions*, and therefore form an abstraction of concrete traces treated as symbolic objects and constraint solving.

Example 6.1. Let us consider again the simplified model of BAC of Example 2.2. When executing the passport process $P(k, n)$ until reaching the success token ok, the constraints aggregate as

$$C = X_0 \vdash^? x_0 \ \wedge\ x_0 =^? \textsc{get\_challenge} \ \wedge$$
$$X \vdash^? x \ \wedge\ \mathsf{sdec}(x, k) =^? n$$

The constraint solver will gradually deduce that solutions to this constraint need to map x to a term of the form $\mathsf{senc}(y_1, y_2, y_3)$, and will add the equations $y_1 =^? n$ and $y_3 =^? k$. △

Partition tree To decide trace equivalence between two plain processes P and Q, the procedure underlying DeepSec builds a refined tree of symbolic executions of P and Q, called a *partition tree.* This finite, symbolic tree intuitively embodies all scenarios of (potential violations of) equivalence, and the final decision criterion is a simple syntactic check on this tree.

More technically, nodes of the partition tree contain sets of symbolic processes derived from P or Q; that is, a branch is a symbolic abstraction of a subset of $\mathbb{T}(P) \cup \mathbb{T}(Q)$. It is constructed in a way that each node contains all—and only—equivalent processes reachable from P or Q with given trace actions tr. When generating this tree, trace equivalence holds if and only if each node contains at least one symbolic process derived from P and one from Q.

6.2 Symbolic matching

Subprocess matchings To make the integration into DeepSec easier, we used an alternative characterisation of equivalence by session that is closer to trace equivalence. In essence, it expresses the structural constraints imposed by twin processes as explicit bijections between labels (as defined in Section 4.1) that we call *session matchings.* A precise definition is given in Appendix A, with a proof that this is equivalent to the twin-process-based definition.

In practice, our implementation consists of keeping track of these session matchings into the nodes of the partition tree generated by DeepSec. The set of all these bijections is then updated at each new symbolic transition step in the partition tree, among others to satisfy the requirement that matched subprocesses should have the same skeleton.

Example 6.2. Consider two initial processes

$$P = c(x).P_0 \mid c(x).P_1 \mid \overline{c}\langle u\rangle.P_2$$
$$Q = c(x).Q_0 \mid \overline{c}\langle u'\rangle.Q_1 \mid c(x).Q_2\,.$$

In the root of the partition tree, P and Q will be labeled by 0, i.e. the root will contain the two symbolic processes

$$(\{[\,P\,]^0\}, \varnothing, \varnothing) \qquad\qquad (\{[\,Q\,]^0\}, \varnothing, \varnothing)\,.$$

There is only a single bijection between their labels, i.e. the identity $0 \mapsto 0$. Upon receiving this initial node, DeepSec applies the symbolic transition corresponding to our rule (Par), hence generating the two symbolic processes

$$(\{\!\{[\,c(x).P_0\,]^{0.1};\ [\,c(x).P_1\,]^{0.2}\ ;\ [\,\overline{c}\langle u\rangle.P_2\,]^{0.3}\}\!\}, \varnothing, \varnothing)$$
$$(\{\!\{[\,c(x).Q_0\,]^{0.1};\ [\,\overline{c}\langle u'\rangle.Q_1\,]^{0.2};\ [\,c(x).Q_2\,]^{0.3}\}\!\}, \varnothing, \varnothing)$$

There are then only two possible bijection of labels that respect the skeleton requirement of twin processes:

$$\begin{array}{lcl} 0.1 \mapsto 0.1 & & 0.1 \mapsto 0.3 \\ 0.2 \mapsto 0.3 & \text{and} & 0.2 \mapsto 0.1 \\ 0.3 \mapsto 0.2 & & 0.3 \mapsto 0.2 \end{array}$$ △

These bijections are kept within the node of the partition tree and updated along side the other transformation rules of DeepSec. For obvious performance reasons, we cannot represent them by a naive enumeration of all process permutations. Fortunately, the skeleton requirement ensures an invariant that the set S of session matchings between two processes A and B is always of the form

$$S = \{\pi \mid \forall i, \forall \ell \in C_i, \pi(\ell) \in D_i\}$$

where the sets $C_1, \ldots, C_n$ form a partition of the labels of A and $D_1, \ldots, D_n$ a partition of the labels of B. In particular, S can succinctly be stored as a simple association list of equivalence classes.

Decision of equivalence Finally, as our trace refinements depend on two sets $\mathbb{O}^\forall$ and $\mathbb{O}^\exists$, we annotate each symbolic process in the node by $\forall$, $\exists$ or $\forall\exists$ tags. They mark whether the trace from the root of the partition tree to the tagged process is determined to be in $\mathbb{O}^\forall$, $\mathbb{O}^\exists$ or both respectively. For instance, the two initial symbolic processes in the root of the partition tree are labeled by $\forall\exists$. We also provide a decision procedure for inclusion by session $\sqsubseteq_s$ that consists of tagging one of the initial processes as $\forall$ and the other one as $\exists$.

The decision criterion for equivalence is then strenghtened. For equivalence to hold, not only each node of the partition tree should contain at least one process originated from P and one process originated from Q, but each of them that has the tag $\forall$ should be paired with at least one other process of the node with the tag $\exists$.

6.3 Integration

From a high-level of abstraction, the integration of the universal optimisations described in sections Sections 4 and 5 prune some branches of the partition tree—those that abstract traces that do not belong to $\mathbb{O}^\forall_{\mathsf{c+r+}s}$. For instance in Section 4.2, we showed that to prove equivalence by session, we can always perform non-input actions in priority. Therefore on a process $\bar{c}\langle u\rangle.P \mid c(x).Q$, we prevent DeepSec from generating a node corresponding to the execution of the input due to the presence of the output.

The integration of other optimisations is more technical in a symbolic setting, in particular the *lexicographic reduction* $\mathbb{O}^\forall_{\mathsf{c+r}}$ described in Section 4.6. Remember that it discards traces that do not satisfy the predicate Minimal, that identifies lexicographically-minimal traces among those obtained by permutation of independent blocks. Unfortunately, the definition of independence (Definition 4.1) is only defined for ground actions—and not their symbolic counterpart, that intuitively abstracts a set of ground actions. A branch may therefore be removed only if *all* its solutions violate the predicate Minimal. However, by Proposition 3.8, it is correct to only partially implement such optimisations.

7 EXPERIMENTS

In practice Based on the high-level description of the previous section, we extended the implementation of DeepSec to decide equivalence by session of P and Q. Upon completing an analysis, two cases can arise:

(1) The two processes are proved equivalent by session. Then they are also trace equivalent by Proposition 3.1.
(2) The two processes are not equivalent by session and DeepSec returns an attack trace t, say, in P, as a result.

In the second case, when using equivalence by session as a heuristic for trace equivalence, the conclusion is not straightforward. As discussed in Section 3.2, the witness trace t may not violate trace equivalence (false attack). We integrated a simple test to our prototype, that checks whether this is the case or not. For that we leverage the internal procedure of DeepSec by, intuitively, restricting the generation of the partition tree for checking $P \sqsubseteq_{tr} Q$ to the unique branch corresponding to the trace t.

If this trace t appears to violate trace equivalence, which is the case for example in our analysis of two sessions of the BAC protocol, we naturally conclude that $P \not\approx_{tr} Q$. Otherwise, the false attack may guide us to discover a real attack: our analysis of session equivalence consider traces with a specific shape (see Sections 4 and 5). Thus, we implemented a simple heuristic that, whenever a false attack is discovered, also checks whether different permutations of actions of this false attack could lead to a true attack. For instance, this heuristic allowed us to disprove trace equivalence in some analyses of $n \geqslant 3$ sessions of BAC. When our heuristic cannot discover a true attack, the result is not conclusive: the processes may well be trace equivalent or not. We leave to future work the design of a complete decision procedure for trace equivalence that builds on a preliminary analysis of equivalence by session.

Experimental setting We report experiments (Figure 4) comparing the scope and efficiency of the following two approaches for proving trace equivalence:

- The original version of DeepSec as a baseline;
- The analysis leveraging our contributions (preliminary analysis of equivalence by session, test of false attack if it fails, and then the heuristic attempting to reconstruct a true attack).

We describe the benchmarks below in more details. The column **# roles** is an indicator of the intricacy of the system (number of parallel processes that the model file exhibits).

Benchmarks were carried out on 20 Intel Xeon 3.10GHz cores, with 50 Gb of memory. We ran the toy example described in this paper on a single core to illustrate simply the algorithmic improvements compared to DeepSec. As DeepSec supports parallelisation, we distributed the computation of the other, bigger proofs over 20 cores. The implementation and the specification files are available at `https://deepsec-prover.github.io/`.

Running example: toy BAC We modelled the simplified analysis of unlinkability in the BAC protocol described in Examples 2.2 and 2.4 as a simple instance to compare our prototype and DeepSec in terms of scope and efficiency. We gather several variants:

- 2 sessions: both DeepSec and our prototype are able to find an attack trace
- 3 sessions: DeepSec times out and our prototype finds a false attack. This is due to the fact that, by executing outputs in priority (recall the por in Section 4), more intermediate actions are available to match the trace. However our heuristic manages to reconstruct a true attack trace by delaying some output actions.
- we also consider a variant where we remove the GET_CHALLENGE message from the protocol description. Our prototype now reports a false attack for 3 sessions and fails to conclude.

The failure in the last variant is not a limitation of our heuristic: by pushing the limits of the baseline version of DeepSec, we actually obtained that trace equivalence held (about 8 days of computation on a version of DeepSec prior to 2.0.0). This is intuitively

Protocol	scenario	# roles	DeepSec baseline	DeepSec eq. by session
Toy BAC	2 identical	4	↯ <1s	↯ <1s
	2 identical + 1 fresh	6	⏱	↯ <1s
Toy BAC no GET_CHALLENGE	2 identical	4	↯ <1s	↯ <1s
	2 identical + 1 fresh	6	(✓) ⏱	✗ <1s
BAC	1 identical + 1 fresh	4	↯ <1s	↯ <1s
	2 identical + 1 fresh	6	⏱	↯ <1s
	3 identical + 1 fresh	8	⏱	↯ <1s
	2 identical + 2 fresh	8	⏱	✓ 19s
	4 identical + 1 fresh	10	⏱	↯ 2s
	3 identical + 2 fresh	10	⏱	↯ 9m31s
	2 identical + 3 fresh	10	⏱	✓ 2h56m
Helios vote swap	no revote	6	✓ <1s	✓ <1s
	$2 \times$ A $1 \times$ B	11	✓ 7min29	✓ 6s
	$3 \times$ A $1 \times$ B	12	✓ 1h38m	✓ 14s
	$3 \times$ A $2 \times$ B	13	⏱	✓ 36s
	$4 \times$ A $2 \times$ B	14	⏱	✓ 1m20s
	$7 \times$ A $3 \times$ B	18	⏱	✓ 17m54
Helios BPRIV	2 honest + 1 dishonest 7 ballots (19 scenarios)	9 (each)	⏱	✓ 1m14s (total)
Scytl	vote privacy	5	✓ 29s	✓ <1s
AKA	anonymity	8	✓ <1s	✓ <1s

✓ trace equivalence verified ↯ trace equivalence violated ⏱ timeout (12 hours)
✗ false attack (disproves session equivalence but unable to conclude for trace equivalence)

Figure 4: Experimental evaluation (DeepSec v2.0.0, Apr. 2020)

because the attacker cannot statically distinguish between a fresh nonce n (as output by passports) and a cipher $\mathrm{senc}(n', r, k)$ (as output by readers). In particular, without the GET_CHALLENGE, each passport can perform at toplevel an output action that is indistinguishable from a reader output, leaving much more possibilities for matching traces.

On the contrary if we assume that the adversary can distinguish between passport and reader actions (which is achieved in the model by using distinct channels for the passport and the reader processes), our prototype manages to disprove trace equivalence.

BAC We also studied a more realistic model of BAC [For04]. The baseline version of DeepSec still fails to analyse 3 or more sessions, while our prototype reaches up to 5 sessions. On one side of the equivalence all n systems are distinct (fresh), while on the other side a same system may appear several times: our analysis indicates that, depending on the precise setting, the security property may be violated in the model or not. This is due to the error codes raised when a passport communicates with a wrong reader: depending on how many identical systems the process contains, the same number of errors may not be observable.

Although not present in the result table, we also implemented inclusion by session (see Section 6.2) as it is sometimes used to define other flavours of unlinkability.

Helios We also consider the Helios protocol for electronic voting [Adi08]. We analyse vote privacy of a version that uses zero-knowledge proofs to ensure the voter knows the plaintext of her vote, thus avoiding copy-attacks [CS13]. Vote privacy is formalised using a classical vote-swapping model, that is, we want to prove the equivalence of two situations where two honest votes have been exchanged.

A reduction result of Arapinis et al. [ACK16] ensures that, for such models, it is sufficient to consider two honest voters and one dishonest voter (that is implicit in the model, embedded in the intruder capabilities) to obtain a proof of the system for an unbounded number of sessions. Such scenarios could already be handled by automated analysers, e.g. DeepSec [CKR18a]. However, when revoting is allowed, as it is the case for Helios, one needs to consider all scenarios when the tally accepts 7 ballots. In particular, it is not sufficient to consider only re-votes by the adversary, but also arbitrary revotes of the two honest voters. In Figure 4 we listed several scenarios, indexed by how many times the honest voters A and B are sending revotes.

This kind of analysis is out of the scope of many automated analysers. For example, Figure 4 shows that DeepSec fails to prove after 12h of computation any scenarios where more than one honest revote is emitted. In [ACK16] the ProVerif proofs are limited to dishonest revotes. We compiled several intermediary scenarios to give an overview of the verification-time growth using our prototype, but all are subsumed by the last scenario were we allow A to revote 7 times and B 3 times. Indeed, using a simple symmetry argument on A and B this covers all scenarios where honest voters cast a total of 7 ballots. Note however that, strictly speaking, the reduction result of [ACK16] does not bound the number of *emitted* honest revotes (that may not be effectively received by the ballot box) that have to be considered during an analysis of vote privacy; extensions of this reduction should be considered in the future.

We also experimented an other model of voting privacy inspired by the game-based definition BPRIV [BCG+15]. In this definition the (re)votes are dicted to honest voters by the adversary, which permits to effectively model revotes of arbitrary values. As reported in Figure 4 the prototype handles the 19 queries modelling all revote scenarios for 7 emitted ballots, in a total of a few minutes.

About the modelling of mixnets. The version of Helios we analyse relies on a mixnet, which can be represented in several ways that may trigger or not a false attack. Mixnets are usually modelled as processes receiving the values to mix, and then outputing them in an arbitrary order induced by the inherent non-determinism of concurrency. However this can be performed using two models (where $c \in Ch_{priv}$):

$$\mathsf{MixSeq} = c(x).\, c(y).\, (\overline{c}\langle x\rangle \mid \overline{c}\langle y\rangle)$$
$$\mathsf{MixPar} = (c(x).\, \overline{c}\langle x\rangle) \mid (c(y) \mid \overline{c}\langle y\rangle)$$

In the second case, subprocess-matching constraints arise earlier in the trace, triggering a false attack. However, the natural modelling of MixSeq allows to complete a security proof. We observed the same behaviour on other experimentation on voting protocols with mixnets.

Other case studies As side experiments, we also tried our prototype on other model files of similar tools that we could find in the literature. We performed for example an analysis of vote privacy of an e-voting protocol by Scytl deployed in the Swiss canton of Neuchâtel, based on the ProVerif file presented in [CGT18]. We also studied anonymity in a model of the AKA protocol deployed in 3G telephony networks [AMR+12] (without XOR), presented in the previous version of DeepSec [CKR18a].

8 CONCLUSION AND FUTURE WORK

In this paper we introduce a new process equivalence, the equivalence by session. We show that it is a sound proof technique for trace equivalence which allows for several optimisations when performing automated verification. This includes powerful partial order reductions, that were previously restricted to the class of determinate processes, and allows to exploit symmetries that naturally arise when verifying multiple sessions of a same protocol. In addition to the theoretical basis we have implemented these techniques in the DeepSec tool and evaluated their effectiveness in practice. The optimisations indeed allowed for efficient verification of non-determinate processes that were previously out of scope of existing techniques.

We also discussed how to handle the false attacks, that are a natural consequence of the fact that equivalence by session is a strict refinement of trace equivalence. We implemented a test to verify automatically, when equivalence by session is disproved, whether the underlying attack is genuine with respect to trace equivalence. When this is not the case, as part of future work it would be interesting to refine the part of the proof that failed, while exploiting that some parts of the system has already been shown to satisfy equivalence.

REFERENCES

[ABF18] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 2018.

[ACK16] Myrto Arapinis, Véronique Cortier, and Steve Kremer. When are three voters enough for privacy properties? In *European Symposium on Research in Computer Security (ESORICS)*, 2016.

[Adi08] Ben Adida. Helios: web-based open-audit voting. In *Conference on Security symposium (SS)*, 2008.

[AMR+12] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *IEEE Symposium on Logic in Computer Science (LICS)*, 2005.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming*, 2008.

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

[BCG+15] David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi. A comprehensive analysis of game-based ballot privacy definitions. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[BDH14] David Baelde, Stéphanie Delaune, and Lucca Hirschi. A reduced semantics for deciding trace equivalence using constraint systems. In *International Conference on Principles of Security and Trust (POST)*, 2014.

[BDH15] David Baelde, Stéphanie Delaune, and Lucca Hirschi. Partial order reduction for security protocols. *International Conference on Concurrency Theory (CONCUR)*, 2015.

[BDH18a] David Baelde, Stéphanie Delaune, and Lucca Hirschi. POR for security protocol equivalences - beyond action determinism. In *European Symposium on Research in Computer Security (ESORICS)*, 2018.

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

[BDS15] David A. Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[CB13] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with proverif. In *Proceedings of the 2nd International Conference on Principles of Security and Trust (POST'13)*, 2013.

[CCCK16] Rohit Chadha, Vincent Cheval, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. *ACM Transactions on Computational Logic*, 2016.

[CD09] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *IEEE Computer Security Foundations Symposium (CSF)*, 2009.

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

[CDSV04] Ivan Cibrario, Luca Durante, Riccardo Sisto, and Adriano Valenzano. Exploiting symmetries for testing equivalence in the spi calculus. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2004.

[CGT18] Véronique Cortier, David Galindo, and Mathieu Turuani. A formal analysis of the neuchâtel e-voting protocol. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018.

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

[CJM03] Edmund Clarke, Somesh Jha, and Will Marrero. Efficient verification of security protocols using partial-order reductions. *STTT*, 2003.

[CKR18a] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. DEEPSEC: Deciding Equivalence Properties in Security Protocols – Theory and Practice. In *IEEE Symposium on Security and Privacy (S&P)*, 2018.

[CKR18b] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. The DEEPSEC prover. In *International Conference on Computer Aided Verification (CAV)*, 2018.

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

[CS13] Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. *Journal of Computer Security*, 2013.

[DSV03] Luca Durante, Riccardo Sisto, and Adriano Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2003.

[DY81] Danny Dolev and Andrew C. Yao. On the security of public key protocols. In *Symposium on Foundations of Computer Science (FOCS)*, 1981.

[ES96] E Allen Emerson and A Prasad Sistla. Symmetry and model checking. *Formal methods in system design*, 1996.

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

[MVB10] Sebastian Mödersheim, Luca Viganò, and David A. Basin. Constraint differentiation: Search-space reduction for the constraint-based analysis of security protocols. *Journal of Computer Security*, 2010.

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

[TNH16] Alwen Tiu, Nam Nguyen, and Ross Horne. SPEC: an equivalence checker for security protocols. In *Asian Symposium on Programming Languages and Systems (APLAS)*, 2016.

A EXPLICIT SESSION MATCHINGS

In this section we present an alternative characterisation of equivalence by session. The process matchings modelled by twin processes are here represented by an explicit permutation with properties mirroring the structure of twin processes. This formalisation makes closer link with the definition of trace equivalence and is more convenient convenient for some proofs (see e.g. Appendices B and D). Besides, note that this is the characterisation we use in the implementation, fitting better to the existing procedure of the DeepSec prover for trace equivalence.

Session matchings We first characterise the condition under which, given two traces t, t', there exists t^2 such that $\mathsf{fst}(t^2) = t$ and $\mathsf{snd}(t^2) = t'$. For that we rely on the notion of *labels* introduced in Section 4.1 to make reference to subprocess positions. In the rest of the paragraph, we refer to two labelled extended processes A_0, B_0 such that $\mathsf{skel}(A_0) = \mathsf{skel}(B_0)$, and as usual we assume that neither of them contains two labels such that one is the prefix of the other. We also let $t \in \mathbb{T}(P), t' \in \mathbb{T}(Q)$

$$t : A_0 \xrightarrow{[\,a_1\,]^{\ell_1}} \cdots \xrightarrow{[\,a_n\,]^{\ell_n}} A_n \qquad t' : B_0 \xrightarrow{[\,b_1\,]^{\ell'_1}} \cdots \xrightarrow{[\,b_n\,]^{\ell'_n}} B_n$$

We write L and L' the sets of labels appearing in t and t'.

Definition A.1 . A *session matching* for t and t' is a bijection $\pi : L \to L'$ verifying the following properties

(1) for all $i \in [\![0, n]\!]$, π defines a bijection from the labels of A_i to the labels of B_i. Besides if A_i and B_i contain processes $[\,P\,]^\ell$ and $[\,Q\,]^{\pi(\ell)}$ respectively then $\mathsf{skel}(P) = \mathsf{skel}(Q)$
(2) for all $i \in [\![1, n]\!]$, $\pi(\ell_i) = \ell'_i$
(3) $\forall \ell \cdot p \in dom(\pi), \exists q, \pi(\ell \cdot p) = \pi(\ell) \cdot q$

Proposition A.1 . The following two points are equivalent:

(1) $tr(t) = tr(t')$ (labelled removed) and there exists a session matching for t and t'.
(2) $\exists t^2,\ \mathsf{fst}(t^2) = t$ and $\mathsf{snd}(t^2) = t'$.

Besides in the i^{th} extended twin process of t^2, the process labelled ℓ in A_i is paired with the process labelled $\pi(\ell)$ in B_i.

Proof of (1) ⇒ (2). The trace t^2 can be easily constructed by induction on the length of t:

- Item (1) of Definition A.1 ensure that the twin processes in t^2 are composed of pairs of processes with the same skeleton,
- Item (2) ensures that pairs of transitions of P and Q can be mapped into transitions of twin-processes, and
- The permutations that are required by applications of the rule (Match) can be inferred from Item (3). Indeed, consider two instances of the rule (Par) in t and t':

$$(\{\!\!\{[\,P_1 \mid \ldots \mid P_n\,]^\ell\}\!\!\} \cup \mathcal{P}, \Phi) \xrightarrow{\tau} (\{\!\!\{[\,P_i\,]^{\ell \cdot i}\}\!\!\}_{i=1}^n \cup \mathcal{P}, \Phi)$$

$$(\{\!\!\{[\,Q_1 \mid \ldots \mid Q_n\,]^{\ell'}\}\!\!\} \cup \mathcal{Q}, \Psi) \xrightarrow{\tau} (\{\!\!\{[\,Q_i\,]^{\ell' \cdot i}\}\!\!\}_{i=1}^n \cup \mathcal{Q}, \Psi)$$

Given π a session matching for t and t', we consider the permutation of $[\![1, n]\!]$ mapping $i \in [\![1, n]\!]$ to the (unique) j such that $\pi(\ell \cdot p) = \ell' \cdot j$. This permutation can be used to construct the instance of rule (Match) corresponding to these two (Par) transitions. ◻

Proof of (2) ⇒ (1). Let t^2 be a trace given by Item (2). We lift the labellings of $t = \mathsf{fst}(t^2)$ and $t' = \mathsf{snd}(t^2)$ to the twin processes appearing in t^2. Then as explicited in the theorem statement it sufficies to consider π the mapping from L to L' such that $\pi(\ell) = \ell'$ for all twin process $([\,P\,]^{\ell}, [\,Q\,]^{\ell'})$ appearing in t^2. A quick induction on the length of t^2 shows that π is well defined and is a session matching for t and t'. We recall in particular the invariant that each A_i or B_i only contains labels that are incomparable w.r.t. the prefix ordering. ◻

Important properties As a direct corollary, we give an alternative characterisation of equivalence by session.

Proposition A.2 . Let P, Q be plain processes in $\rightsquigarrow$-normal form such that $\mathsf{skel}(P) = \mathsf{skel}(Q)$. The following points are equivalent:
(1) $P \sqsubseteq_s Q$
(2) for all $t \in \mathbb{T}(P)$, there exist $t' \in \mathbb{T}(Q)$ and a session matching for t and t' such that $t \sim t'$ (*note:* in "$t \sim t'$" the comparison of $tr(t)$ and $tr(t')$ does not take the labels into account)

Besides the relation $\sim_s$ on traces defined by $t \sim_s t'$ *iff* there exists t^2 such that $\mathsf{fst}(t^2) = t$ and $\mathsf{snd}(t^2) = t'$, is an equivalence relation. More precisely:

Proposition A.3 . For all traces t_1, t_2, t_3:
(1) id is a session matching for t_1 and t_1
(2) if π is a session matching for t_1 and t_2 then π^{-1} is a session matching for t_2 and t_1
(3) if π is a session matching for t_1 and t_2, and π' for t_2 and t_3, then $\pi' \circ \pi$ is a session matching for t_1 and t_3

B FALSE ATTACKS AND DETERMINACY

In this section we give a detailed proof of the claim of Section 3 that false attacks cannot arise for determinate processes, i.e.:

Proposition 3.2 . If P, Q are determinate plain processes such that $P \approx_{tr} Q$ then $P \approx_s Q$.

In the proof, by slight abuse of notation, we may say that an extended process is determinate. We also cast the notion of skeleton to extended processes by writing

$$\mathsf{skel}((\mathcal{P}, \Phi)) = \mathsf{skel}(\mathcal{P}) = \textstyle\bigcup_{P \in \mathcal{P}} \mathsf{skel}(P)\,,$$

and to traces with

$$\mathsf{skel}(A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n) = \mathsf{skel}(A_0) \cdot \mathsf{skel}(A_1) \cdot \ldots \cdot \mathsf{skel}(A_n)\,.$$

That is, the skeleton of a trace is the sequence of the skeletons of the processes of which it is composed. Thus, if

$$t : A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n \qquad t' : B_0 \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_p} B_p$$

we have $\mathsf{skel}(t) = \mathsf{skel}(t')$ *iff* $n = p$ and for all $i \in [\![0, n]\!]$, $\mathsf{skel}(A_i) = \mathsf{skel}(B_i)$.

Simplifying equivalence First we simplify the problem by forcing the application of (Par) rules in priority in traces.

Definition B.1 . If P is a plain process in $\rightsquigarrow$-normal form, we write $\mathbb{T}_\tau(P)$ the set of traces where the rule (Par) is always performed in priority, i.e. where the rules (In) and (Out) are never applied to extended processes $(\mathcal{P}, \Phi)$ such that $\mathcal{P}$ contains a process with a parallel a its root (i.e. a process P such that $|\mathsf{skel}(P)| > 1$).

Proposition B.1 . If P, Q are plain processes in $\rightsquigarrow$-normal form such that $\mathsf{skel}(P) = \mathsf{skel}(Q)$:
- $P \sqsubseteq_{tr} Q$ *iff* $\forall t \in \mathbb{T}_\tau(P), \exists t' \in \mathbb{T}_\tau(Q), t \sim t'$
- $P \sqsubseteq_s Q$ *iff* $\forall t \in \mathbb{T}_\tau(P), \exists t^2 \in \mathbb{T}(P, Q), t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$

Proof. The first point is standard. The proof of the second point can be seen as a corollary of the compression optimisations of equivalence by session (see Section 4.2). ◻

Definition B.2 . An extended process $A = (\{\!\{P_1, \ldots, P_n\}\!\}, \Phi)$ is τ-*deterministic* if there is at most one $i \in [\![1, n]\!]$ such that P_i has a parallel operator at its root (i.e. $|\mathsf{skel}(P_i)| > 1$).

The τ-determinism will be an invariant in proofs by induction on the length of traces. More precisely, if A, B are extended processes we call $\mathsf{Inv}(A, B)$ the property stating
(i) A_i, B_i are determinate
(ii) $\mathsf{skel}(A_i) = \mathsf{skel}(B_i)$
(iii) $A_i \sim B_i$
(iv) A_i, B_i are τ-deterministic, and A_i contains a process with a parallel operator at its root (i.e. a process P_i such that $|\mathsf{skel}(P_i)| > 1$) *iff* B_i does.

Equivalence and inclusion We prove that trace equivalence coincides with a notion of trace inclusion strengthened with identical actions and skeleton checks.

Proposition B.2 . If P, Q are determinate plain processes in $\rightsquigarrow$-normal form s.t. $\mathsf{skel}(P) = \mathsf{skel}(Q)$, then the following points are equivalent
(1) $P \approx_{tr} Q$
(2) $\forall t \in \mathbb{T}_\tau(P),\ \exists t' \in \mathbb{T}_\tau(Q),\ \begin{cases} tr(t) = tr(t') \\ \Phi(t) \sim \Phi(t') \\ \mathsf{skel}(t) = \mathsf{skel}(t') \end{cases}$

Proof of (2) ⇒ (1). Given A, B two determinate extended processes we write $\varphi(A, B)$ the property stating that

$$\forall t \in \mathbb{T}_\tau(A),\ \exists t' \in \mathbb{T}_\tau(B),\ \begin{cases} tr(t) = tr(t') \\ \Phi(t) \sim \Phi(t') \\ \mathsf{skel}(t) = \mathsf{skel}(t') \end{cases}.$$

Note that $\varphi(A, B)$ implies $\mathsf{skel}(A) = \mathsf{skel}(B)$ by choosing the empty trace. In particular, to prove (2) ⇒ (1), it sufficies to prove that for all A, B determinate, $\varphi(A, B) \Rightarrow A \sqsubseteq_{tr} B$ and $\varphi(A, B) \Rightarrow \varphi(B, A)$.

The first implication is immediate. As for the second, we prove that for all extended processes A_0, B_0 such that $\varphi(A_0, B_0)$

and $\mathsf{Inv}(A_0, B_0)$, and all

$$t' : B_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B_n \quad \in \mathbb{T}_\tau(B_0)\,,$$

there exists

$$t : A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n \quad \in \mathbb{T}_\tau(A_0)\,,$$

s.t. for all $i \in [\![0, n]\!]$, $\mathsf{Inv}(A_i, B_i)$. This is sufficient to conclude as $\mathsf{Inv}(P, Q)$ holds for any determinate plain processes P, Q in $\rightsquigarrow$-normal form s.t. $\mathsf{skel}(P) = \mathsf{skel}(Q)$.

We proceed by induction on n. If $n = 0$ the conclusion is immediate. Otherwise, assume by induction hypothesis that it holds for any trace of length $n - 1$.

▹ *case 1: $\alpha_1 = \tau$.*

We know that B_0 does not contain private channels by determinacy ($\mathsf{Inv}(A_0, B_0)$ Item (i)). Therefore, the transition $B_0 \xrightarrow{\tau} B_1$ is derived by the rule (Par). In particular by $\mathsf{Inv}(A_0, B_0)$ Item (iv), there also exists a transition $A_0 \xrightarrow{\tau} A_1$. The conclusion can now follow from the induction hypothesis applied to A_1, B_1; but to apply it we have to prove that $\varphi(A_1, B_1)$ and $\mathsf{Inv}(A_1, B_1)$ hold.

$\longrightarrow$ *proof that* $\varphi(A_1, B_1)$.

Let $s \in \mathbb{T}_\tau(A_1)$. Then $(A_0 \xrightarrow{\tau} A_1)\cdot s \in \mathbb{T}_\tau(A_0)$ and by $\varphi(A_0, B_0)$ there exists $(B_0 \xrightarrow{\tau} B_1') \cdot s' \in \mathbb{T}_\tau(B_0)$ such that $tr(s) = tr(s')$, $\Phi(t) \sim \Phi(t')$ and $\mathsf{skel}(t) = \mathsf{skel}(t')$. But by τ-determinism of B_0 we deduce that $B_1 = B_1'$, and $s' \in \mathbb{T}_\tau(B_1)$ satisfies the expected requirements.

$\longrightarrow$ *proof that* $\mathsf{Inv}(A_1, B_1)$.

(i) A_0 and B_0 are determinate and determinacy is preserved by transitions.
(ii) $\mathsf{skel}(A_1) = \mathsf{skel}(A_0) = \mathsf{skel}(B_0) = \mathsf{skel}(B_1)$
(iii) $A_0 \sim B_0$ and the rule (Par) does not affect the frame.
(iv) A_0 and B_0 are τ-deterministic and τ-determinism is preserved by transitions (w.r.t. $\mathbb{T}_\tau$). Besides due to the $\rightsquigarrow$-normalisation, we know that neither of A_1 nor B_1 contain a parall operator, hence the result.

▹ *case 2: $\alpha_1 \neq \tau$.*

By definition $\mathbb{T}_\tau(B_0)$, we know that the rule (Par) is not applicable to B_0; neither to A_0 by $\mathsf{Inv}(A_0, B_0)$ Item (iv), which means that traces of $\mathbb{T}_\tau(B_0)$ may start by an application of rules (In) or (Out). Using this and the fact that $\mathsf{skel}(A_0) = \mathsf{skel}(B_0)$ ($\mathsf{Inv}(A_0, B_0)$ Item (ii)), we obtain that there exists a transition $A_0 \xrightarrow{\alpha_1} A_1$. The conclusion can now follow from the induction hypothesis applied to A_1, B_1; but to apply it we have to prove that $\varphi(A_1, B_1)$ and $\mathsf{Inv}(A_1, B_1)$ hold.

$\longrightarrow$ *proof that* $\varphi(A_1, B_1)$.

The argument is the same as its analogue in *case 1*, using the determinacy of B_0 instead of its τ-determinism.

$\longrightarrow$ *proof that* $\mathsf{Inv}(A_1, B_1)$.

(i) A_0 and B_0 are determinate and determinacy is preserved by transitions.
(ii) By applying $\varphi(A_0, B_0)$ with the trace $t_0 : A_0 \xrightarrow{\alpha_1} A_1$, we obtain a trace $t_0' : B_0 \xrightarrow{\alpha_1} B_1'$ such that $\mathsf{skel}(A_1) = \mathsf{skel}(B_1')$. But by determinacy of B_0, the transition $B_0 \xrightarrow{\alpha_1} B_1$ is the only transition from B_0 that has label α_1, hence $B_1 = B_1'$ and the conclusion.
(iii) Identical proof as that of Item (ii) above, using the fact that $A_1 \sim B_1'$ instead of $\mathsf{skel}(A_1) = \mathsf{skel}(B_1')$.
(iv) Let us write

$$A_0 = (\{\!\!\{P_0\}\!\!\} \cup \mathcal{P}, \Phi) \qquad A_1 = (\{\!\!\{P_1\}\!\!\} \cup \mathcal{P}, \Phi')$$
$$B_0 = (\{\!\!\{Q_0\}\!\!\} \cup \mathcal{Q}, \Psi) \qquad B_1 = (\{\!\!\{Q_1\}\!\!\} \cup \mathcal{Q}, \Psi')$$

As we argued already at the beginning of *case 2*, neither $\mathcal{P}$ nor $\mathcal{Q}$ contain processes with parallel operators at their roots. Therefore, we only have to prove that P_1 has a parallel operator at its root *iff* Q_1 does. For cardinality reasons, this a direct corollary of the following points:

– $\mathsf{skel}(P_0) = \mathsf{skel}(Q_0)$ (same action α_1 being executable at topelevel),
– $\mathsf{skel}(A_0) = \mathsf{skel}(B_0)$ (hypothesis $\mathsf{Inv}(A_0, B_0)$), and
– $\mathsf{skel}(A_1) = \mathsf{skel}(B_1)$ (Item (ii) proved above). □

Proof of (1) ⇒ (2). The proof will follow in the steps as the other implication (we construct the trace t' by induction on the length of t while maintaining the invariant Inv).

More formally, we prove that for all extended processes A_0, B_0 such that $A_0 \approx_{tr} B_0$ and $\mathsf{Inv}(A_0, B_0)$, and all

$$t : A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n \quad \in \mathbb{T}_\tau(A_0)\,,$$

there exists

$$t' : B_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} B_n \quad \in \mathbb{T}_\tau(B_0)\,,$$

s.t. for all $i \in [\![0, n]\!]$, $\mathsf{Inv}(A_i, B_i)$.

We proceed by induction on n. We proceed by induction on n. If $n = 0$ the conclusion is immediate. Otherwise, assume by induction hypothesis that it holds for any trace of length $n - 1$.

▹ *case 1: $\alpha_1 = \tau$.*

Similarly to the converse implication, there exists a transition $B_0 \xrightarrow{\tau} B_1$ (derived by (Par)) and it sufficies to prove that $A_1 \approx_{tr} B_1$ and $\mathsf{Inv}(A_1, B_1)$ hold in order to apply the induction hypothesis and conclude.

$\longrightarrow$ *proof that* $A_1 \approx_{tr} B_1$.

Let $s \in \mathbb{T}_\tau(A_1)$. Then $(A_0 \xrightarrow{\tau} A_1) \cdot s \in \mathbb{T}_\tau(A_0)$ and since $A_0 \approx_{tr} B_0$ there is $(B_0 \xrightarrow{\tau} B_1') \cdot s' \in \mathbb{T}_\tau(B_0)$ such that

$$(A_0 \xrightarrow{\tau} A_1) \cdot s \sim (B_0 \xrightarrow{\tau} B_1') \cdot s'\,.$$

But by τ-determinism of B_0 we deduce that $B_1 = B_1'$, and thus $s' \in \mathbb{T}_\tau(B_1)$ and $s \sim s'$. This justifies that $A_1 \sqsubseteq_{tr} B_1$, and a symmetric argument can be used for the converse inclusion $B_1 \sqsubseteq_{tr} A_1$.

$\longrightarrow$ *proof that* $\mathsf{Inv}(A_1, B_1)$.

By the exact same arguments as that of the analogue case in the converse implication.

▹ *case 2: $\alpha_1 \neq \tau$.*

Similarly to the converse implication, there exists a transition $B_0 \xrightarrow{\alpha_1} B_1$ and it sufficies to prove that $A_1 \approx_{tr} B_1$ and $\mathsf{Inv}(A_1, B_1)$ hold in order to apply the induction hypothesis and conclude.

$\rightarrow$ *proof that $A_1 \approx_{tr} B_1$.*

The argument is the same as its analogue in *case 1*, using the determinacy of B_0 instead of its τ-determinism.

$\rightarrow$ *proof that* $\mathsf{Inv}(A_1, B_1)$.

This is the proof obligation whose arguments substantially differ from that of the converse implication.

(i) A_0 and B_0 are determinate and determinacy is preserved by transitions.

(ii) We assume by contradiction that $\mathsf{skel}(A_1) \neq \mathsf{skel}(B_1)$. By symmetry, say that $\mathsf{skel}(A_1) \nsubseteq \mathsf{skel}(B_1)$ and let $s \in \mathsf{skel}(A_1) \smallsetminus \mathsf{skel}(B_1)$. By definition of $\mathbb{T}_\tau(A_0)$, we know that the rule (Par) is neither applicable to A_0 nor B_0; in particular, there exists a transition $A_1 \xrightarrow{\alpha} A$ derived from rule (In) or (Out) (the one corresponding to the skeleton s) such that $B_1 \not\xrightarrow{\alpha}$.
But by determinacy of B_0, the transition $B_0 \xrightarrow{\alpha_1} B_1$ is the only transition from B_0 that has label α_1. Thus, this yields a contradiction with $A_0 \approx_{tr} B_0$: more precisely the trace $A_0 \xrightarrow{\alpha_1} A_1 \xrightarrow{\alpha} A$ is not matched.

(iii) By determinacy of B_0, the transition $t'_0 : B_0 \xrightarrow{\alpha_1} B_1$ is the only transition from B_0 that has label α_1. In particular, using the hypothesis $A_0 \approx_{tr} B_0$, we obtain that $t'_0 \in \mathbb{T}_\tau(B_0)$ is the only trace such that
$$t_0 : (A_0 \xrightarrow{\alpha_1} A_1) \sim t'_0 .$$
In particular $A_1 \sim B_1$.

(iv) Same cardinality argument as the analogue case in the converse implication. □

Session matchings Proposition B.2 is the core result of the proof. We now connect it with the equivalence by session by using the characterisation of Appendix A.

Proposition B.3 . Let P, Q two determinate plain processes in $\rightsquigarrow$-normal form and two labelled traces $t \in \mathbb{T}_\tau(A_0)$ and $t' \in \mathbb{T}_\tau(Q)$ such that $tr(t) = tr(t')$ and $\mathsf{skel}(t) = \mathsf{skel}(t')$. Then there exists a session matching for t and t'.

Proof. We prove that for all τ-deterministic, determinate extended processes A_0 and B_0, and
$$t : A_0 \xrightarrow{[\,\alpha_1\,]^{\ell_1}} \cdots \xrightarrow{[\,\alpha_n\,]^{\ell_n}} A_n \qquad t' : B_0 \xrightarrow{[\,\alpha_1\,]^{\ell'_1}} \cdots \xrightarrow{[\,\alpha_n\,]^{\ell'_n}} B_n$$
if $\mathsf{skel}(t) = \mathsf{skel}(t')$, then there exists a session matching for t and t'. We proceed by induction on n. If $n = 0$ the session matching is $\pi : \varepsilon \mapsto \varepsilon$. Otherwise, let us write
$$A_{n-1} = (\{\!\{ [\,P\,]^{\ell_n} \}\!\} \cup \mathcal{P}, \Phi) \qquad B_{n-1} = (\{\!\{ [\,Q\,]^{\ell'_n} \}\!\} \cup \mathcal{Q}, \Psi)$$
By induction hypothesis, let π be a session matching for the first $n-1$ transitions of t and t'; in particular, the labels of A_{n-1} are in the domain of π.

▹ *case 1: $\alpha_n \neq \tau$.*

In this case we write
$$A_n = (\{\!\{ [\,P'\,]^{\ell_n} \}\!\} \cup \mathcal{P}, \Phi') \qquad B_n = (\{\!\{ [\,Q'\,]^{\ell'_n} \}\!\} \cup \mathcal{Q}, \Psi')$$
First of all, we observe that $\mathsf{skel}(P) = \mathsf{skel}(Q)$ because the same observable action α_n can be performed at the root of P and Q. In particular, by determinacy (hypothesis), unicity of the process with a given label (invariant of the labelling procedure), and Item (1) of Definition A.1, we deduce that $\pi(\ell_n) = \ell'_n$.

Therefore by the hypothesis $\mathsf{skel}(A_{n-1}) = \mathsf{skel}(B_{n-1})$, we obtain $\mathsf{skel}(\mathcal{P}) = \mathsf{skel}(\mathcal{Q})$. Hence $\mathsf{skel}(P') = \mathsf{skel}(Q')$ by the hypothesis $\mathsf{skel}(A_n) = \mathsf{skel}(B_n)$. All in all, π is a session matching for the whole traces t and t'.

▹ *case 2: $\alpha_n = \tau$.*

In this case we write
$$P = P_1 \mid \cdots \mid P_k \qquad A_n = (\{\!\{ [\,P_i\,]^{\ell_n \cdot i} \}\!\}_{i=1}^{k} \cup \mathcal{P}, \Phi')$$
$$Q = Q_1 \mid \cdots \mid Q_{k'} \qquad B_n = (\{\!\{ [\,Q_i\,]^{\ell'_n \cdot i} \}\!\}_{i=1}^{k'} \cup \mathcal{Q}, \Psi')$$
Since determinacy excludes private channels, the last transition of t and t' is derived from the rule (Par). By τ-determinism, this means that P and Q are the only processes in A_{n-1} and B_{n-1}, respectively, that contain a parallel operator at their roots. In particular, by Item (1) of Definition A.1, we deduce that $\pi(\ell_n) = \ell'_n$ and $\mathsf{skel}(P) = \mathsf{skel}(Q)$; and thus $k = k'$.

Therefore, there exists a permutation σ of $[\![1, k]\!]$ such that for all $i \in [\![1, k]\!]$, $\mathsf{skel}(P_i) = \mathsf{skel}(Q_{\sigma(i)})$ (although this is not needed for the proof, this permutation appears to be unique by determinacy). Thus if $\pi' : L \rightarrow L'$ is the function extending π and such that
$$\forall i \in [\![1, k]\!], \pi'(\ell \cdot i) = \pi(\ell) \cdot \sigma(i) ,$$
then π' is a session matching for t and t'. □

Altogether Propositions A.2 to B.3 justify the following corollary (that actually appears to be stronger than Proposition 3.2).

Corollary B.4 . If P and Q are determinate plain processes in $\rightsquigarrow$-normal form, $P \approx_{tr} Q$ *iff* $P \sqsubseteq_s Q$.

C CORRECTNESS OF POR

In this section we prove the results presented in Section 4.

C.1 Permutability of independent actions

We give the proof of the core correctness argument, namely that traces can be considered up to permutation of independent actions (Proposition 4.3). First we prove it for traces of two actions.

Proposition C.1 . If $\alpha \parallel \beta$ and $t : A \xRightarrow{\alpha\beta} B$, then there exists a trace $u : A \xRightarrow{\beta\alpha} B$. It has the property that for all traces $u^2 : A^2 \xRightarrow{\beta\alpha}_s B^2$ such that $\mathsf{fst}(u^2) = u$, there exists $t^2 : A^2 \xRightarrow{\alpha\beta}_s B^2$ such that $\mathsf{fst}(t^2) = t$.

Proof. Since the labels of α and β are incomparable w.r.t. the prefix ordering by independence, the trace t needs have the form

$$A = (\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R}, \Phi) \xrightarrow{\alpha} (\mathcal{P}' \cup \mathcal{Q} \cup \mathcal{R}, \Phi') \xrightarrow{\beta}_s (\mathcal{P}' \cup \mathcal{Q}' \cup \mathcal{R}, \Phi'')$$

with $(\mathcal{P}, \Phi) \xrightarrow{\alpha} (\mathcal{P}', \Phi')$ and $(\mathcal{Q}, \Phi') \xrightarrow{\beta} (\mathcal{Q}', \Phi'')$. Now we construct the trace u, by a case analysis on α and β. In each case, we omit the construction of the trace t^2 that can be inferred easily.

- ▹ *case 1:* α and β are inputs or τ actions.

In particular $\Phi'' = \Phi' = \Phi$ and it sufficies to choose

$$u : (\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R}, \Phi) \xrightarrow{\beta} (\mathcal{P} \cup \mathcal{Q}' \cup \mathcal{R}, \Phi) \xrightarrow{\alpha} (\mathcal{P}' \cup \mathcal{Q}' \cup \mathcal{R}, \Phi)\,.$$

- ▹ *case 2:* α is an output and β is an input or a τ action.

In particular $\Phi'' = \Phi' = \Phi \cup \{\mathsf{ax} \mapsto m\}$ with $\mathsf{ax} \notin dom(\Phi)$ and ax does not appear in β. Then it sufficies to choose the trace

$$u : (\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R}, \Phi) \xrightarrow{\beta} (\mathcal{P} \cup \mathcal{Q}' \cup \mathcal{R}, \Phi) \xrightarrow{\alpha} (\mathcal{P}' \cup \mathcal{Q}' \cup \mathcal{R}, \Phi')\,.$$

- ▹ *case 3:* α is an input or a τ action and β is an output.

Similar to case 2.

- ▹ *case 4*: α and β are both outputs.

Then $\Phi' = \Phi \cup \{\mathsf{ax} \mapsto m\}$ and $\Phi'' = \Phi' \cup \{\mathsf{ax}' \mapsto m'\}$ with $\mathsf{ax} \neq \mathsf{ax}'$, $\{\mathsf{ax}, \mathsf{ax}'\} \cap dom(\Phi) = \emptyset$. Then we choose

$$\begin{aligned} u : (\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R}, \Phi) &\xrightarrow{\beta} (\mathcal{P} \cup \mathcal{Q}' \cup \mathcal{R}, \Phi \cup \{\mathsf{ax}' \mapsto m'\}) \\ &\xrightarrow{\alpha} (\mathcal{P}' \cup \mathcal{Q}' \cup \mathcal{R}, \Phi'')\,. \end{aligned}$$

□

Then Proposition 4.3 can be obtained by induction on the hypothesis of π permuting independent actions of tr, using Proposition C.1. We actually prove the stronger result:

Proposition C.2. If $t : A \overset{\mathsf{tr}}{\Longrightarrow} B$ and π permutes independent actions of tr, then $A \overset{\pi.\mathsf{tr}}{\Longrightarrow} B$. This trace is unique if we take labels into account, and is referred as $\pi.t$. It has the property that for all $u^2 : A^2 \overset{\pi.\mathsf{tr}}{\Longrightarrow}_s B^2$ such that $\mathsf{fst}(u^2) = \pi.t$, there exists $t^2 : A^2 \overset{\mathsf{tr}}{\Longrightarrow}_s B^2$ such that $\mathsf{fst}(t^2) = t$.

Proof. The uniqueness of $\pi.t$ is immediate, as a quick induction on the length of traces shows that any labelled trace u is uniquely determined by the action word $tr(u)$ (labels included). We then construct $\pi.t$ by induction on the hypothesis that π permutes independent actions of $tr(t)$. Let us write

$$t : A = A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n = B\,.$$

If $\pi = \mathrm{id}$ it sufficies to choose $\pi.t = t$. Otherwise let us write $\pi = \pi_0 \circ (i\ \ i{+}1)$ with $\alpha_i \parallel \alpha_{i+1}$ and π_0 permutes independent actions of $\mathsf{tr}' = \alpha_p \cdots \alpha_{i-1}\alpha_{i+1}\alpha_i\alpha_{i+2} \cdots \alpha_n$. By Proposition C.1, there exists a trace

$$u : A_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{i-1}} A_{i-1} \xRightarrow{\alpha_{i+1}\alpha_i} A_{i+1} \xrightarrow{\alpha_{i+2}} \cdots \xrightarrow{\alpha_n} A_n$$

such that for all $u^2 : A^2 \overset{\mathsf{tr}'}{\Longrightarrow}_s B^2$ verifying $\mathsf{fst}(u^2) = u$, there exists $t^2 : A^2 \overset{\mathsf{tr}}{\Longrightarrow}_s B^2$ such that $\mathsf{fst}(t^2) = t$. Then since π_0 permutes independent actions of $\mathsf{tr}' = tr(u)$, it sufficies to choose $\pi.t = \pi_0.u$ by induction hypothesis. □

Then we can easily extend this result to $\equiv_{\mathsf{por}}$.

Proposition C.3. Let $t : A \overset{\mathsf{tr}}{\Longrightarrow} B$ be a trace and $t' \equiv_{\mathsf{por}} t$. Then writing $tr(t') = \mathsf{tr}'$ we have $t' : A \overset{\mathsf{tr}'}{\Longrightarrow} B$ and, for all $u^2 : A^2 \overset{\mathsf{tr}'}{\Longrightarrow} B^2$ such that $t' = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$, there exists $t^2 : A^2 \overset{\mathsf{tr}}{\Longrightarrow} B^2$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$.

Proof. For the sake of reference, let us write $H(t, t')$ the property to prove. We reason by induction on the hypothesis $t \equiv_{\mathsf{por}} t'$.

- ▹ *case 1*: $t' = \pi.t$, π permutes independent actions of t.

Direct consequence of Proposition C.2.

- ▹ *case 2*: t' is recipe-equivalent to t.

Let u^2 with $t' = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$. By static equivence, for any recipes ξ_1, ξ_2 such that

$$\xi_1\Phi(\mathsf{fst}(u^2)) =_E \xi_2\Phi(\mathsf{fst}(u^2))\,,$$

we also have

$$\xi_1\Phi(\mathsf{snd}(u^2)) =_E \xi_2\Phi(\mathsf{snd}(u^2))\,.$$

In particular, t^2 can be obtained by operating on the second component of u^2 the same recipe transformations that have been operated to transform $t' = \mathsf{fst}(u^2)$ into t.

- ▹ *case 3*: (transitivity) $H(t, s)$ and $H(s, t')$ for some trace s.

Let u^2 with $t' = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$. By hypothesis $H(s, t')$ there exists s^2 such that $s = \mathsf{fst}(s^2) \sim \mathsf{snd}(s^2)$. Hence the result by hypothesis $H(t, s)$. □

And finally we have the Proposition 4.4 that is a corollary of this result.

Proposition 4.4 (correctness of por). Let $\mathbb{O}_1^{\forall} \subseteq \mathbb{O}_2^{\forall}$ be universal optimisations. We assume that for all $t \in \mathbb{O}_2^{\forall}$, there exists t_{ext} such that t is a prefix of t_{ext}, and $t' \in \mathbb{O}_1^{\forall}$ such that $t' \equiv_{\mathsf{por}} t_{ext}$. Then $\mathbb{O}_1^{\forall}$ is a correct refinement of $\mathbb{O}_2^{\forall}$.

Proof. Let $\approx_i = \sqsubseteq_i \cap \sqsupseteq_i$ the notion of equivalence induced by $\mathbb{O}_i^{\forall}$. The inclusion $\approx_2 \subseteq \approx_1$ is immediate. Let us then assume $P \sqsubseteq_1 Q$ and prove $P \sqsubseteq_2 Q$. Let $t \in \mathbb{T}(P) \cap \mathbb{O}_2^{\forall}$. Without loss of generality, we assume t maximal, i.e. that there are no transitions possible from its last process. Therefore by hypothesis, there exists $t' \equiv_{\mathsf{por}} t$ such that $t' \in \mathbb{O}_1^{\forall}$. Since $P \sqsubseteq_1 Q$, there is $u^2 \in \mathbb{T}(P, Q)$ such that

$$t' = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)\,.$$

Therefore by Proposition C.3, there exists $t^2 \in \mathbb{T}(P, Q)$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$. □

C.2 Additional results

We provide some utility results on independent permutations of actions. First, about composition of permutations:

Proposition C.4. Let t be a trace, π permuting independent actions of t, and π' permuting independent actions of $\pi.t$. Then $\pi.\pi'.t = (\pi \circ \pi').t$.

Proof. By definition, if $t : A \stackrel{\text{tr}}{\Longrightarrow} B$, $\pi.\pi'.t$ is the unique trace of the form $A \stackrel{\pi.\pi'.\text{tr}}{\Longrightarrow} B$, and $(\pi \circ \pi').t$ is the unique trace of the form $A \stackrel{(\pi\circ\pi').\text{tr}}{\Longrightarrow} B$. Hence the result since $\pi.\pi'.\text{tr} = (\pi \circ \pi').\text{tr}$ by definition of a group action. ◻

This formalises that the group-action properties of $(\pi, \text{tr}) \mapsto \pi.\text{tr}$ carry on to traces. Then, we also discuss the domain extension of permutations. If π is a permutation of $[\![1, n]\!]$, we define π^{+q}_{+p} permutation of $[\![1, n + p + q]\!]$ by

$$\pi^{+q}_{+p}(x) = \begin{cases} p + \pi(x - p) & \text{if } p < x \leqslant n + p \\ x & \text{otherwise} \end{cases}$$

in particular, the following result is immediate:

Proposition C.5 (extension). If π permutes independent actions of v, $\pi^{+|w|}_{+|u|}$ permutes independent actions of uvw.

C.3 Decomposition into phases

In this section we prove correct the refinement at the very basis of our partial-order reductions, namely that all traces can be decomposed into phases (modulo permutation of independent actions).

Proposition 4.5. $\mathbb{O}^{\forall}_{\text{c},b}$ is a correct refinement of $\mathbb{O}^{\forall}_{\text{all}}$.

Proof. By Proposition 4.4, it sufficies to prove that for all traces t that are maximal (i.e. whose last process is irreducible), there exists π permuting independent actions of t such that $\pi.t$ can be decomposed into phases.

We prove this by induction on the length of t. If t is empty the result is immediate: π is the identity and the phase decomposition consists of a unique empty negative block. Otherwise let us write

$$t : (A \xrightarrow{\alpha} B) \cdot t' .$$

Note in particular that the trace t' is also maximal. By induction hypothesis, there exists π' permuting independent actions of t' such that

$$\pi'.t' = b^-_0 \cdot b^+_1 \cdot b^-_1 \cdot b^+_2 \cdot b^-_2 \cdots b^+_n \cdot b^-_n$$

where each b^+_i is a positive or null phase, and each b^-_i is a negative phase.

▹ *case 1*: α is an output or a parallel action.

Then $(A \xrightarrow{\alpha} B) \cdot b^-_0$ is a negative phase and it sufficies to choose $\pi = (\pi')^{+0}_{+1}$.

▹ *case 2*: $\alpha = [\,\tau\,]^{\ell_1|\ell_2}$ (internal communication).

Let us write E the multiset of actions of the word $\text{tr}(b^-_0)$. We partition it into $E = F \uplus G$ where

$$F = \{\!\{\beta \in E \mid \alpha \parallel \beta\}\!\}$$
$$G = \{\!\{[\,a\,]^{\ell} \in E \mid \ell_1 \preccurlyeq_{pref} \ell \text{ or } \ell_2 \preccurlyeq_{pref} \ell\}\!\}$$

where $\preccurlyeq_{pref}$ refers to the prefix ordering on words. This is indeed a partition of E thanks to the invariant that any label appearing in t' is either incomparable with ℓ_1 and ℓ_2, or a suffix of ℓ_1 or ℓ_2. For the same reason, all actions in F are independent of all actions in G: it is therefore straightforward to construct π^-_0 permuting independent actions of b^-_0 such that

$$\pi^-_0.b^-_0 : B \stackrel{\text{tr}_F\cdot\text{tr}_G}{\Longrightarrow} C \qquad \text{tr}_F \in F^\star \qquad \text{tr}_G \in G^\star .$$

Then, we let σ permuting independent actions of

$$s = (A \xrightarrow{\alpha} B) \cdot (\pi^-_0.b^-_0)$$

such that $\sigma.s = A \stackrel{\text{tr}_F}{\Longrightarrow} B' \xrightarrow{\alpha} B'' \stackrel{\text{tr}_G}{\Longrightarrow} C$. By definition of F and G, we have $polar(B') \neq \infty$. And by the hypothesis that b^-_0 is a negative phase, its last process C has not a polarity of $-\infty$ neither. Therefore $A \stackrel{\text{tr}_F}{\Longrightarrow} B'$ and $B'' \stackrel{\text{tr}_G}{\Longrightarrow} C$ are negative phases. All in all, it sufficies to choose

$$\pi = \sigma^{+p}_{+0} \circ (\pi^-_0)^{+p}_{+1} \circ (\pi')^{+0}_{+1} \qquad \text{with } p = \sum_{i=1}^{n} |b^+_i| + |b^-_i|$$

▹ *case 3*: $\alpha = [\,c(\xi)\,]^{\ell}$.

Let us write

$$A = (\{\!\{[\,c(x).P\,]^{\ell}\}\!\} \cup \mathcal{P}, \Phi) \qquad B = (\{\!\{[\,P'\,]^{\ell}\}\!\} \cup \mathcal{P}, \Phi)$$

If the label ℓ does not appear in $tr(t')$, then by maximality of t it needs be that $polar(P') = 0$ and $(A \xrightarrow{\alpha} B)$ is therefore a positive phase. In particular $\pi'.t$ is already decomposed into phases and it sufficies to choose $\pi = (\pi')^{+0}_{+1}$.

Otherwise assume that ℓ appears in $tr(t')$. We write

$$\text{tr}^-_i = tr(b^-_i) \qquad \text{tr}^+_i = tr(b^+_i)$$

We also consider the phase of t' in which the first action of P' is executed, i.e. the first phase b such that ℓ appears in $tr(b)$. Note that, thanks to the invariant that any label appearing in t' is either incomparable or a suffix of ℓ, α is independent of all actions of all phases of t' preceding b.

▹ *case 3a*: $b = b^+_i$ is a positive or null phase.

Then we fix σ permuting independent actions of

$$\alpha \cdot \text{tr} \qquad \text{with tr} = \text{tr}^-_0 \cdot \text{tr}^+_1 \cdot \text{tr}^-_1 \cdots \text{tr}^+_{i-1} \cdot \text{tr}^-_{i-1}$$

such that, writing $s = (A \xrightarrow{\alpha} B) \cdot b^-_0 \cdot b^+_1 \cdot b^-_1 \cdots b^+_{i-1} \cdot b^-_{i-1}$,

$$\sigma.s : A \stackrel{\text{tr}}{\Longrightarrow} A' \xrightarrow{\alpha} A'' .$$

If $b = b^+_i$ is a null phase, $A' \xrightarrow{\alpha} A''$ is a positive phase. If b is a positive phase, $(A' \xrightarrow{\alpha} A'') \cdot b$ is a positive phase too. In both cases, it sufficies to choose

$$\pi = \sigma^{+p}_{+0} \circ (\pi')^{+0}_{+1} \qquad \text{with } p = \sum_{j=i}^{n} |b^+_j| + |b^-_j|$$

▹ *case 3b*: $b = b^-_i$ is a negative phase.

Similarly to case 2, we fix E the multiset of actions appearing in the word tr^-_i and we partition it as $E = F \uplus G$

$$F = \{\beta \in E \mid \alpha \parallel \beta\} \qquad G = \{[\,a\,]^{\ell'} \in E \mid \ell \preccurlyeq_{pref} \ell'\} .$$

And again we let π_i^- permuting tr_i^- such that

$$\pi_i^-.b_i^- : R \xrightarrow{\mathsf{tr}_F} S \xrightarrow{\mathsf{tr}_G} T \qquad \mathsf{tr}_F \in F^\star \qquad \mathsf{tr}_G \in G^\star .$$

Then we let σ permuting independent actions of

$$\alpha \cdot \mathsf{tr} \cdot \mathsf{tr}_F \qquad \text{with } \mathsf{tr} = \mathsf{tr}_0^- \cdot \mathsf{tr}_1^+ \cdot \mathsf{tr}_1^- \cdots \mathsf{tr}_i^+$$

such that, writing $s = (A \xrightarrow{\alpha} B) \cdot b_0^- \cdot b_1^+ \cdot b_1^- \cdots b_i^+ \cdot (\pi_i^-.b_i^-)$,

$$\sigma.s : A \xrightarrow{\mathsf{tr}} A' \xrightarrow{\mathsf{tr}_F} A'' \xrightarrow{\alpha} S \xrightarrow{\mathsf{tr}_G} T .$$

For the same reason as in case 2, $A' \xrightarrow{\mathsf{tr}_F} A''$ and $S \xrightarrow{\mathsf{tr}_G} T$ are negative phases. It therefore sufficies to choose

$$\pi = \sigma_{+0}^{+p} \circ (\pi_i^-)_{1+|\mathsf{tr}|}^{p} \circ (\pi')_{+1}^{+0} \quad \text{with } p = \sum_{j=i+1}^{n} |b_j^+| + |b_j^-| \quad \square$$

C.4 Improper positive phases

In this section we prove the correctness of the optimisation $\mathbb{O}^{\forall}_{\mathsf{c+i}}$ consisting of delaying improper blocks as much as possible in traces, as introduced in Section 4.3. First we prove that improper blocks are essentially independent of all blocks following them.

Proposition C.6 . Let $t \in \mathbb{O}^{\forall}_{\mathsf{c}}$ of the form $t = b \cdot u$ where $u \in \mathbb{O}^{\forall}_{\mathsf{c}}$ and b is an improper block. Then there exists u' recipe equivalent to u such that b is independent of all blocks of u'.

Proof. By definition of improper blocks Item (2), if $A \xrightarrow{\mathsf{tr}} (\mathcal{P}, \Phi)$ is improper then for all $[\,P\,]^L \in \mathcal{P}$ such that $P \neq 0$, the labels of L are prefix-incomparable with all labels of tr. In particular a straightforward induction shows that all actions of b are sequentially independent of all actions of u. Besides let us write by definition

$$b : (\mathcal{P}, \Phi) \xrightarrow{\mathsf{tr}} (Q, \Phi \cup \{\mathsf{ax}_1 \mapsto t_1, \ldots, \mathsf{ax}_n \mapsto t_n\})$$

with ξ_i a recipe such that $\xi_i \Phi =_E t_i$. We then let the substitution

$$\sigma = \{\mathsf{ax}_1 \mapsto \xi_1, \ldots, \mathsf{ax}_n \mapsto \xi_n\}$$

as well as u' the trace obtained by replacing in u all actions of the form $c(\xi)$ by $c(\xi\sigma)$. By definition, u is recipe equivalent to u' and, since no axioms introduced in b appear in u', b is data independent of all blocks of u'. All in all, b is independent of all blocks of u'. $\square$

The main argument is then a substantially-simple but technical induction delaying improper blocks one by one. For that we prove the following auxiliary result about the preservation of (im)properness when permuting independent blocks, since it is not the case in general as discussed in Section 4.4.

Proposition C.7 . Let a sequence of blocks $t = b_1 \cdots b_n$ and a transposition $\pi = (i \;\; i{+}1)$ for some $i \in [\![1, n-1]\!]$. We assume that $b_i \parallel b_{i+1}$. Then

(1) *Delaying an improper block preserves improperness:* if b_i is improper then the $i{+}1^{\text{th}}$ block of $\pi.t$ is also improper.
(2) *Advancing a proper block preserves properness:* if b_{i+1} is proper then the i^{th} block of $\pi.t$ is also proper.
(3) *Swapping two improper blocks preserve improperness:* if b_i and b_{i+1} are improper then the i^{th} and $i{+}1^{\text{th}}$ blocks of $\pi.t$ are improper.

Proof. Item (1) follows from the fact that for any ground term t, recipe ξ and frame Φ, if $\xi\Phi =_E t$ then $\xi(\Phi \cup \Phi') =_E t$ for all frames Φ' such that $dom(\Phi) \cap dom(\Phi') = \varnothing$. Item (2) follows from the fact that for any ground term t and frames Φ, Φ' such that $dom(\Phi) \cap dom(\Phi') = \varnothing$, if t is not deducible in $\Phi \cup \Phi'$ then it cannot be deducible in Φ neither. Let us detail more the argument for Item (3), by writing

$$b_i : (\mathcal{P}_0, \Phi_0) \xrightarrow{\mathsf{tr}} (\mathcal{P}_1, \Phi_0 \cup \Phi_1)$$
$$b_{i+1} : (\mathcal{P}_1, \Phi_0 \cup \Phi_1) \xrightarrow{\mathsf{tr}'} (\mathcal{P}_2, \Phi_0 \cup \Phi_1 \cup \Phi_2)$$

If we write b'_i, b'_{i+1} the i^{th} and $i{+}1^{\text{th}}$ blocks of $\pi.t$, respectively, they have the form

$$b'_i : (Q_0, \Phi_0) \xrightarrow{\mathsf{tr}'} (Q_1, \Phi_0 \cup \Phi_2)$$
$$b'_{i+1} : (Q_1, \Phi_0 \cup \Phi_2) \xrightarrow{\mathsf{tr}} (Q_2, \Phi_0 \cup \Phi_1 \cup \Phi_2) .$$

To show that b'_i and b'_{i+1} are improper, it sufficies to verify the Item (3) of the definition since the two other items immediately follow from the fact that b_i and b_{i+1} are improper.

▹ *Proof that b'_i is improper.*

Let $t \in im(\Phi_2)$ and let us construct a recipe ξ such that $\xi\Phi_0 =_E t$. Since b_{i+1} is improper there exists a recipe ξ_0 such that $\xi_0(\Phi_0 \cup \Phi_1) =_E t$. Besides b_i is also improper hence for all $\mathsf{ax} \in dom(\Phi_1)$ there exists a recipe ξ_{ax} such that $\xi_{\mathsf{ax}}\Phi_0 =_E \mathsf{ax}\,\Phi_1$. It thus sufficies to choose

$$\xi = \xi_0\{\mathsf{ax} \mapsto \xi_{\mathsf{ax}} \mid \mathsf{ax} \in dom(\Phi_1)\} .$$

▹ *Proof that b'_{i+1} is improper.*

If $t \in im(\Phi_2)$, since b_i is improper there exists a recipe ξ such that $\xi\Phi_0 =_E t$. In particular $\xi(\Phi_0 \cup \Phi_1) =_E t\Phi_1 =_E t$. $\square$

Proposition 4.9 . For all $t \in \mathbb{O}^{\forall}_{\mathsf{c}}$, there exists $t' \equiv_{\mathsf{b\text{-}por}} t$ such that $t' \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ and t' has the same number of improper blocks t.

Proof. Let us decompose t in blocks as $t : b_0^- \cdot b_1 \cdots b_n$. We say that $i \in [\![1, n-1]\!]$ is a *pending index* in t when b_i is an improper block and there exists $j > i$ such that b_j is a proper block. We prove the proposition by induction on the number of pending indexes.

If there are no pending indexes then $t \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ by definition. Otherwise let p be the minimal pending index in t. By Proposition C.6 there exists v' recipe equivalent to $b_{p+1} \cdots b_n$ such that b is independent of all blocks of v'. Thus the permutation

$$\pi = (n \;\; n{-}1) \circ (n{-}1 \;\; n{-}2) \circ \cdots \circ (p{+}2 \;\; p{+}1) \circ (p{+}1 \;\; p)$$

permutes independent blocks of $s = b_1 \cdots b_p \cdot v'$. Let us prove that $\pi.s$ has less pending indexes than t, and as many improper blocks as t. Once this is established this will conclude the proof: this makes it possible apply the induction hypothesis to $\pi.s$, thus obtaining $t' \equiv_{\mathsf{b\text{-}por}} \pi.s$ such that $t' \in \mathbb{O}^{\forall}_{\mathsf{c+i}}$ and t' has as many improper blocks as $\pi.s$. In particular $t' \equiv_{\text{b-por}} \pi.s \equiv_{\text{b-por}} s \equiv_{\text{b-por}} t$ hence the desired conclusion.

Let us thus compare the number of pending indexes and improper blocks of t and $\pi.s$. First of all, (im)properness is preserved under recipe equivalence, and t therefore has as many pending indexes and improper blocks as s. Let us decompose s and $\pi.s$ in blocks as

$$s = b_0^- \cdot b_1^s \cdots b_n^s \qquad \pi.s = b_0^- \cdot b_1^{\pi.s} \cdots b_n^{\pi.s}$$

We then let $i \in [\![1, n]\!]$.

▹ *case 1: $i < p$. Let us prove that b_i and $b_i^{\pi.s}$ are proper (in particular i is not a pending index in $\pi.s$).*

This follows from the fact that $b_i^{\pi.s} = b_i$ and b_i is proper (otherwise p would not be the minimal pending index in t).

▹ *case 2: $i = n$. Let us prove that $b_n^{\pi.s}$ is an improper block.*

Since $b_p^s = b_p$ is improper by hypothesis, $n - p$ applications of Proposition C.7 Item (1) show that the $b_p^{\pi.s}$ is improper.

▹ *case 3: $p \leqslant i < n$. Let us prove that $b_i^{\pi.s}$ is improper iff b_{i+1}^s is improper.*

Let us write, if $u \in \mathbb{O}_c^{\vee}$, b_i^u the i^{th} block of u. If $a \leqslant b$ we also define the permutation

$$\pi_{a\to b} = (b\ \ b-1) \circ (b-1\ \ b-2) \circ \cdots \circ (a+2\ \ a+1) \circ (a+1\ \ a)$$

with $\pi_{a\to a} = \mathrm{id}$ by convention. As a preliminary result we prove by induction on i that $b_i^{\pi_{p\to i}.s}$ is improper: if $i = p$ we have $b_p^{\pi_{p\to p}.s} = b_p$ which is improper, and if $i > p$ we know that $b_i^{\pi_{p\to i-1}.s}$ is improper by induction hypothesis, hence the result since $\pi_{p\to i} = (i\ \ i-1) \circ \pi_{p\to i-1}$ and by using Proposition C.7 Item (1).

We now prove the actual property, i.e. that $b_i^{\pi.s}$ is improper *iff* b_{i+1}^s is improper. Using the preliminary result above we know that $b_i^{\pi_{p\to i}.s}$ is improper. In particular, using Proposition C.7 Items (2) and (3), we obtain that $b_i^{\pi_{p\to i+1}}.s$ is improper *iff* $b_{i+1}^{\pi_{p\to i}.s} = b_{i+1}^s$ is improper. The conclusion eventually follows from the fact that $b_i^{\pi.s} = b_i^{\pi_{p\to n}.s} = b_i^{\pi_{p\to i+1}.s}$.

▹ *Conclusion: $\pi.s$ contains less pending indexes than s, and the same number of improper blocks.*

The fact that $\pi.s$ and s have the same number of improper blocks follows from the fact that the bijection $\varphi : [\![1, n]\!] \longrightarrow [\![1, n]\!]$ defined by

$$\begin{array}{llll} \varphi(i) = i & \text{if } i < p & \varphi(i) = n & \text{if } i = p \\ \varphi(i) = i+1 & \text{if } p \leqslant i < n & \varphi(i) = p & \text{if } i = n \end{array}$$

verifies $b_i^{\pi.s}$ is improper *iff* $b_{\varphi(i)}^s$ is improper, by the cases 1,2,3.

Let us then show that $\pi.s$ has less pending indexes than s. For that, by case 1, it suffices to prove that $b_p^{\pi.s} \cdots b_n^{\pi.s}$ contains less pending indexes than $b_p^s \cdots b_n^s$. Let $p \leqslant i < n$ such that $b_i^{\pi.s}$ is improper. By case 3, there exists $j \in [\![i+1, n-1]\!]$ such that $b_j^{\pi.s}$ is proper *iff* there exists $j' \in [\![i+2, n]\!]$ such that $b_{j'}^s$ is proper, that is, *iff* $i+1$ is a pending index in s. By case 2 this means that $i < n$ is a pending index in $\pi.s$ *iff* $i + 1$ is a pending index in s. Therefore there are as many pending indexes in $b_p^{\pi.s} \cdots b_n^{\pi.s}$ as in $b_{p+1}^s \cdots b_n^s$, i.e. less than in $b_p^s \cdots b_n^s$ since p is a pending index in s by hypothesis. □

The correctness of the optimisation (Corollary 4.10) then simply follows from this proposition and Corollary 4.8.

C.5 High-priority null phases

In this section we prove the correctness of the optimisation based on executing high-priority internal communications in priority in traces as formalised in Section 4.5. First we state and prove the main technical property of high-priority transitions that makes it correct to execute them as soon as they are available:

Proposition C.8 . Let $d \in Ch_{priv}$ a high-priority channel in A, and a transition $A \xrightarrow{\alpha} B$ that is not an internal communication on d. Then d is high-priority in B.

Proof. We use the definition and thus let a trace of the form

$$B \overset{\mathrm{tr}}{\Longrightarrow} C \qquad \text{no labels of } \mathbb{L}_B(d) \text{ appear in tr},$$

and show that $\mathbb{L}_B(d) = \mathbb{L}_C(d)$. By hypothesis the label of α does not belong to $\mathbb{L}_A(d)$ and therefore, since d is high-priority in A and $\mathbb{L}_A(d) = \mathbb{L}_B(d)$ by definition, we obtain the expected conclusion by considering the trace $A \xrightarrow{\alpha} B \overset{\mathrm{tr}}{\Longrightarrow} C$. □

Using this result we can prove that whenever a channel is high-priority at some point A in a maximal trace, there needs be a later transition that executes an internal communication on this channel that was already available in A.

Proposition C.9 . Let $t \in \mathbb{O}_c^{\vee} \cap \mathbb{T}(A)$ and $d \in Ch_{priv}$ that is high-priority in A. We also assume that t is maximal, i.e. that no transitions are possible at the end of t. Then t can be decomposed into $t = b_0^- \cdot u \cdot b \cdot v$ for some negative phase b_0^- and $u, v \in \mathbb{O}_c^{\vee}$ such that

- b is a block starting with an internal communication on d with action $[\,\tau\,]^{\ell|\ell'}$ with $\{\ell, \ell'\} \subseteq \mathbb{L}_A(d)$
- the block b is independent of all blocks of u.

Proof. We decompose t in phases as follows $t : b_0^- \cdot b_1 \cdots b_n$. We proceed by induction on n the number of blocks of t. Since d is high-priority in A, an internal communication on d is possible in A by definition. In particular since t is maximal, we know that b_1 cannot be the empty trace. Let us therefore perform a case analysis on the first transition of b_1.

▹ *case 1:* t starts with an internal communication on d.

Then it suffices to choose $u = \varepsilon$, $b = b_1$ and $v = b_2 \cdots b_n$.

▹ *case 2:* t starts with a transition that is either a public input or an internal communication on a channel $e \neq d$.

Let us write $t_0 = (b_0^- \cdot b_1) : A_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} A_n$. By applying Proposition C.8 to each transition of t_0, we know that d is also high-priority in each A_i, in particular in A_n. We can therefore apply the induction hypothesis to $b_2 \cdots b_n$. By doing so we get $i \in [\![2, n]\!]$ such that

(1) b_i starts with an internal communication on d with action $[\,\tau\,]^{\ell|\ell'}$ with $\{\ell, \ell'\} \subseteq \mathbb{L}_{A_n}(d)$. Hence $\{\ell, \ell'\} \subseteq \mathbb{L}_A(d)$ since $\mathbb{L}_{A_n}(d) = \mathbb{L}_A(d)$ because d is high-priority in $A_1 = A$. This implies, since b_1 contains no internal communications on d by hypothesis, that the first transition of b_i is sequentially independent—and therefore independent—of all actions of b_1. In particular $b_1 \parallel b_i$.
(2) for all $j \in [\![2, i-1]\!]$, $b_j \parallel b_i$.

Altogether it sufficies to choose $u = b_1 \cdots b_{i-1}$, $b = b_i$ and $v = b_{i+1} \cdots b_n$. □

Proposition 4.14. Let $t = u \cdot v$ a maximal trace, where $u, v \in \mathbb{O}^{\forall}_{\mathsf{c}}$ and v does not contain any high-priority transitions. Then there exists π permuting independent blocks of u such that $\pi.u \in \mathbb{O}^{\forall}_{0}$.

Proof. We decompose u and v in blocks

$$u = b_0^- \cdot b_1 \cdots b_n \qquad\qquad v = b_{n+1} \cdots b_m$$

with $b_i : A_i \xRightarrow{\mathsf{tr}_i} A_{i+1}$. First of all we observe that for all $i > n$, there are no high-priority channels in A_i. Indeed using Proposition C.8, a quick induction on the length of s shows that for all maximal traces $s : A \xRightarrow{\mathsf{tr}} B$ such that there exists a high-priority channel in A, there exists a high-priority transition in s. In the context of our proposition, this would contradict the hypothesis that v does not contain any high-priority transitions. We then prove the property by induction on the number of processes A_i, $i \in [\![1, n]\!]$, such that there is a high-priority channel in A_i.

If there is no such process then it sufficies to choose $\pi = \mathrm{id}$. Otherwise let $p \in [\![1, n]\!]$ be the minimal index such that there exists a high-priority channel in A_p, and $d \in Ch_{priv}$ the minimal such channel w.r.t. $\preccurlyeq_{Ch}$. By Proposition C.9 there exists $q \in [\![p, m]\!]$ such that b_q starts with an internal communication on d and for all $p \leqslant r < q$, $b_r \parallel b_q$. If we consider the minimal such index q, we know by Proposition C.8 that d is high-priority in A_q and in particular $q \leqslant n$ by the preliminary remark. Let us thus assume so. Therefore the permutation

$$\pi_0 = (p \;\; p{+}1) \circ (p{+}1 \;\; p{+}2) \circ \cdots \circ (q{-}2 \;\; q{-}1) \circ (q{-}1 \;\; q)$$

permutes independent blocks of u and the p^{th} block of $\pi_0.u$ starts with a high-priority transition. We write u_p the trace composed of the first p blocks of $\pi_0.u$. We have $u_p \in \mathbb{O}^{\forall}_{0}$: no high-priority transitions are possible in the first $p-1$ blocks of u_p by minimality of p, and the p^{th} block starts with a high-priority transition.

We then let u_{n-p} the remaining $n-p$ blocks of $\pi_0.u$, i.e. the trace such that $\pi_0.u = b_0^- \cdot u_p \cdot u_{n-p}$. We then apply the induction hypothesis to $u_{n-p} \cdot v$ (which is indeed a maximal trace) which gives π_1 permuting independent blocks of u_{n-p} such that $\pi_1.u_{n-p} \in \mathbb{O}^{\forall}_{0}$. It therefore sufficies to choose $\pi = \pi_1{}^{+0}_{+p} \circ \pi_0$ (notation of the extension lemma, Proposition C.5). □

Corollary 4.15. $\mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c+i^*}}$.

Proof. We use the characterisation of Corollary 4.8. Let $t \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ and t_{ext} an arbitrary maximal extension of t (obtained by executing transitions after t as long as it is possible). By Propositions 4.9 and 4.12 there exists $t_{\mathsf{i}^*} \equiv_{\mathsf{b\text{-}por}} t_{ext}$ such that $t_{\mathsf{i}^*} \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$. Let us decompose t_{i^*} as

$$t_{\mathsf{i}^*} = b_0^- \cdot u \cdot v$$

where b_0^- is a negative phase, $u \in \mathbb{O}^{\forall}_{\mathsf{c}}$ only contains proper blocks and $v \in \mathbb{O}^{\forall}_{\mathsf{c}}$ only contains improper blocks. Note that t_{i^*} is maximal since $t_{\mathsf{i}^*} \equiv_{\mathsf{b\text{-}por}} t_{ext}$ and t_{ext} is maximal. Recall also that improper blocks never contain high-priority transitions by definition. There we can apply Proposition 4.14 to obtain π permuting independent actions of u such that $\pi.u \in \mathbb{O}^{\forall}_{0}$. Hence $b_0^- \cdot \pi.u \cdot v \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$, keeping in mind Proposition 4.11. □

C.6 Reduction of independent blocks

Now we prove the correctness of the optimisation $\mathbb{O}^{\forall}_{\mathsf{por}}$ introduced in Section 4.6. We recall that we assume an ordering $\preccurlyeq$ on words of actions such that two words of independent actions are always strictly comparable. In particular we can rephrase more simply the definition of the fact that a block cannot follow an other one:

Proposition C.10. Let two blocks $b \parallel b'$. If b cannot follow b' then $tr(b) \prec tr(b')$, and if b' starts with a high-priority transition then b too and on the same private channel.

Proof. By definition if the block b cannot follow the block b' then (1) $\neg(tr(b') \prec tr(b))$; and (2) if b' starts with a high-priority transition then b too; and (3) b and b' do not start with a high-priority transition on different channels. Hence the result, keeping in mind the assumption on $\preccurlyeq$ that $b \parallel b'$ implies $tr(b) \prec tr(b')$ or $tr(b') \prec tr(b)$. □

We write $\preccurlyeq_{lex}$ the lexicographic extension of $\preccurlyeq$ to traces of same number of blocks, i.e. if $t = b_1 \cdots b_n$ and $t' = b'_1 \cdots b'_n$,

$$t \preccurlyeq_{lex} t' \quad \textit{iff} \quad tr(b_1) \prec tr(b'_1) \;\text{ or }\; \begin{cases} tr(b_1) \preccurlyeq tr(b'_1) \\ b_2 \cdots b_n \preccurlyeq_{lex} b'_2 \cdots b'_n \end{cases}$$

By convention, two traces with a different number of blocks are incomparable w.r.t. $\preccurlyeq_{lex}$. The core argument to prove the correctness of the optimisation is that $\mathbb{O}^{\forall}_{\mathsf{r}}$ contains minimal traces among those obtainable by permutation of independent actions:

Proposition C.11. Let $u = (b_0^- \cdot b_1 \cdots b_n) \in \mathbb{O}^{\forall}_{0}$ and $i \in [\![2, n]\!]$ such that $\neg\mathsf{Minimal}(b_1 \cdots b_{i-1}, b_i)$. We assume i minimal among indexes with this property and write $\pi = (i{-}1 \;\; i)$. Then π permutes independent actions of u, $\pi.u \prec_{lex} u$ and $p \in \mathbb{O}^{\forall}_{0}$ where p consists of the first $i-1$ blocks of $\pi.u$.

Proof. By definition of the predicate $\mathsf{Minimal}$, $b_{i-1} \parallel b_i$, $\mathrm{tr}(b_i) \prec \mathrm{tr}(b_{i-1})$ and b_i cannot follow b_{i-1}. Note that the eventuality $\neg\mathsf{Minimal}(b_1 \cdots b_{i-2}, b_{i-1})$ has been excluded by minimality of i. In particular π permutes independent blocks of u and $\pi.u \prec_{lex} u$. Thus let b'_j the j^{th} block of $\pi.u$, $j \in [\![1, i-1]\!]$ and let us show that it starts with a high-priority transition if one is available. If $j < i-1$ then $\pi(j) = j$, hence $b'_j = b_j$. The conclusion thus follows from the assumption $u \in \mathbb{O}^{\forall}_{0}$. Otherwise $j = i-1$ and we write A such that $b_{i-1} \in \mathbb{T}(A)$. We recall that b_i cannot follow b_{i-1}; therefore by Proposition C.10 we are in one of the following cases.

- ▹ *case 1:* b_{i-1} does not start with a high-priority transition

  In particular the assumption that $u \in \mathbb{O}^{\forall}_{0}$ ensures that no high-priority transitions are possible from A which appears to be the initial process of the block b'_j, hence the conclusion.

- ▹ *case 2:* b_{i-1} and b_i start with a high-priority internal communication on a channel d

  In particular since $u \in \mathbb{O}^{\forall}_{0}$, d is the minimal high-priority channel among those available in A, and the block b'_j (which takes the same transitions as b_i) starts with a high-priority. □

Corollary C.12. Let $t \in \mathbb{O}^{\forall}_{c}$ a maximal trace of the form $t = u \cdot v$ where $u \in \mathbb{O}^{\forall}_{0} \smallsetminus \mathbb{O}^{\forall}_{r}$, and $v \in \mathbb{O}^{\forall}_{c}$ does not contain any high-priority transitions. Then there exists π permuting independent actions of u such that $\pi.u \prec_{lex} u$ and $\pi.u \in \mathbb{O}^{\forall}_{0}$.

Proof. Consider the block decomposition $u = b_0^- \cdot b_1 \cdots b_n$. By hypothesis $u \in \mathbb{O}^{\forall}_{0} \smallsetminus \mathbb{O}^{\forall}_{r}$ and we can therefore fix $i \in [\![2, n]\!]$ the minimal index such that $\neg\mathsf{Minimal}(b_1 \cdots b_{i-1}, b_i)$. By Proposition C.11, $\pi = (i-1 \;\; i)$ permutes independent blocks of u, $\pi.u \prec_{lex} u$ and the first $i-1$ blocks of $\pi.u$ form a trace of $\mathbb{O}^{\forall}_{0}$.

Besides let s consisting of the last $n-i+1$ blocks of $\pi.u$. By hypothesis $s \cdot v$ is maximal and v does not contain high-priority transitions: in particular by Proposition 4.14 there exists σ permuting independent blocks of s such that $\sigma.s \in \mathbb{O}^{\forall}_{0}$. Then we define, using the notations of the extension lemma (Proposition C.5), $\tau = \sigma^{+0}_{+i-1} \circ \pi$. It permutes independent blocks of u, $\tau.u \prec_{lex} u$ and $\tau.u \in \mathbb{O}^{\forall}_{0}$. □

Proposition C.13. Let $t \in \mathbb{O}^{\forall}_{c}$ a maximal trace of the form $t = u \cdot v$ where $u \in \mathbb{O}^{\forall}_{0}$, and $v \in \mathbb{O}^{\forall}_{c}$ does not contain any high-priority transitions. Then there exists π permuting independent blocks of u such that $\pi.u \in \mathbb{O}^{\forall}_{0} \cap \mathbb{O}^{\forall}_{r}$. Besides $\pi.u \prec_{lex} u$ if $u \notin \mathbb{O}^{\forall}_{r}$.

Proof. Let us consider the set of traces

$$U = \mathbb{O}^{\forall}_{0} \cap \{\pi.u \mid \pi \text{ permutes independent blocks of } t\} .$$

The set U is finite (its size is bounded by $|u|!$). In particular the ordering $\preccurlyeq_{lex}$ is well-founded on U, i.e. there exist no infinite decreasing sequences of elements of U w.r.t $\prec_{lex}$. Let us thus show by well-founded induction on u' that for all traces of the form $u' \cdot v$, $u' \in U$, there exists π permuting independent blocks of u' such that $\pi.u' \in \mathbb{O}^{\forall}_{r}$. If $u' \in \mathbb{O}^{\forall}_{r}$ it suffices to choose $\pi = \mathrm{id}$. Otherwise by Corollary C.12 there exists π_0 permuting independent blocks of u' such that $\pi_0.u' \prec_{lex} u'$ and $\pi_0.u' \in U$. Hence the result by induction hypothesis applied to $\pi_0.u'$. □

We can eventually prove the correctness of $\mathbb{O}^{\forall}_{\mathrm{por}}$. It essentially relies on the characterisation of correctness provided by Corollary 4.8, applying two times Proposition C.13. The decreasing argument w.r.t. the lexicographic extension of $\preccurlyeq$ is a simple consequence of the fact that the proof is performed by well founded induction w.r.t. this ordering.

Proposition 4.16. For all maximal traces $t \in \mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$, there exists π permuting independent blocks of t such that $\pi.t \in \mathbb{O}^{\forall}_{\mathrm{por}}$. Besides $\pi.t \prec_{lex} t$ if $t \notin \mathbb{O}^{\forall}_{\mathrm{por}}$, with $\preccurlyeq_{lex}$ the lexicographic extension of $\preccurlyeq$.

Proof. Let us decompose such a trace t into $t : b^- \cdot t_p \cdot t_i$ where b^- is a negative phase, $t_p \in \mathbb{O}^{\forall}_{0}$ only contains proper blocks and $t_i \in \mathbb{O}^{\forall}_{c}$ only contains improper blocks. Besides, by maximality of t_i, no high-priority channels appear in any extended process of t_i (otherwise a block of t_i would start with an internal communication by Proposition C.9, impossible for improper blocks). Therefore $t_i \in \mathbb{O}^{\forall}_{0}$. We can thus apply Proposition C.13 with $u = t_i$ and $v = \varepsilon$ and we obtain π_i permuting independent blocks of t_i such that $\pi_i.t_i \in \mathbb{O}^{\forall}_{0} \cap \mathbb{O}^{\forall}_{r}$ and $\pi_t.t_i \preccurlyeq_{lex} t_i$.

We observe that $\pi_i.t_i$ is also maximal and does not contain any high-priority transitions. By applying Proposition C.13 again, but with $u = t_p$ and $v = \pi_i.t_i$, we obtain π_p permuting independent blocks of t_p such that $\pi_p.t_p \in \mathbb{O}^{\forall}_{0} \cap \mathbb{O}^{\forall}_{r}$ and $\pi_t.t_p \preccurlyeq_{lex} t_p$. In particular, using the notations of the extension lemma (Proposition C.5),

$$\pi = \pi_p{}^{+|t_i|}_{+0} \circ \pi_i{}^{+0}_{+|t_p|}$$

permutes independent actions of t and

$$\pi.t = b_0^- \cdot \pi_p.t_p \cdot \pi_i.t_i \;.$$

There are only improper blocks in $\pi_i.t_i$ by Proposition C.7 Item (3), and therefore $\pi.t \in \mathbb{O}^{\forall}_{\mathrm{c+i^*}}$ by Proposition 4.11. All in all, $\pi.t \in \mathbb{O}^{\forall}_{\mathrm{por}}$ and $\pi.t \preccurlyeq_{lex} t$. □

Corollary 4.17. $\mathbb{O}^{\forall}_{\mathrm{por}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$.

Proof. Let $t \in \mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$ and t_{ext} an arbitrary, maximal extension of t obtained by executing as many transitions of t as possible. By Propositions 4.9, 4.12 and 4.14 there exists $\bar{t} \equiv_{\mathrm{b\text{-}por}} t_{ext}$ such that $\bar{t} \in \mathbb{O}^{\forall}_{\mathrm{c+i^*+0}}$. Then by Proposition 4.16 there exists $u \equiv_{\mathrm{b\text{-}por}} \bar{t}$ such that $u \in \mathbb{O}^{\forall}_{\mathrm{por}}$. Hence the conclusion by Corollary 4.8. □

D CORRECTNESS OF SYMMETRIES

In this section we prove the correctness of the optimisations presented in Section 5, i.e. the reduction by symmetries.

D.1 Preliminary notations and results

First of all we introduce some notions that will be at the core of most proofs of correctness of our reductions by symmetry. Intuitively when there is a symmetry between two processes P_a and P_b, we *relabel* them (that is, we swap their labels) and the resulting process is structurally equivalent to the initial one. Then we use the fact that structurally equivalent processes produce *almost identical* traces, meaning that their can be matched equivalently. We formalise these two notions in the following paragraphs.

Relabelling We first define an action of the group of label permutations. It acts on extended (twin) processes, actions and traces by relabelling. Formally if π is a permutation of labels, we write

$$A\pi \qquad A^2\pi \qquad \alpha\pi \qquad t\pi \qquad t^2\pi$$

respectively the extended process A, extended twin process A^2, labelled action α, trace t and extended twin trace t^2 where each label $\ell \cdot \ell'$ has been replaced by $\pi(\ell) \cdot \ell'$. For this transformation to be well defined, we always consider that π is *consistent* (w.r.t. the object it is applied to) which means that

(1) the set $supp(\pi) = \{\ell \mid \pi(\ell) \neq \ell\}$ only contains pairwise incomparable labels w.r.t. the prefix ordering
(2) if applied to $A = (\{\!\{[P_1]^{\ell_1}, \ldots, [P_n]^{\ell_n}\}\!\}, \Phi)$ then for all $\ell \in supp(\pi)$, there exists $i \in [\![1, n]\!]$ such that ℓ is a prefix of ℓ_i
(3) if applied to $t \in \mathbb{T}(A)$ with $A = (\{\!\{[P_1]^{\ell_1}, \ldots, [P_n]^{\ell_n}\}\!\}, \Phi)$ then π is consistent w.r.t. A
(4) if applied to an extended twin process or an extended twin trace, π is consistent w.r.t. its first projection.

We recall that extended twin processes are only labelled on their first projection. This operation has a lot of trivial but important properties that we will use implicitly in incoming proofs:

Proposition D.1. With the notations above:
(1) $\mathsf{fst}(t^2\pi) = \mathsf{fst}(t^2)\pi$ and $\mathsf{snd}(t^2\pi) = \mathsf{snd}(t^2)$
(2) Hence: $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$ *iff* $t\pi = \mathsf{fst}(t^2\pi) \sim \mathsf{snd}(t^2\pi)$
(3) if $\sigma \in S_n$ and π is the permutation such that $\pi(\ell_i) = \ell_{\sigma(i)}$, $A\pi = \sigma^{-1}.A$.
(4) if X is any of the objects above, $X\mathrm{id} = X$ and $X\pi\pi' = X(\pi' \circ \pi)$.
(5) if π is consistent w.r.t. A, $A \xrightarrow{\alpha} B$ then π is consistent w.r.t. B.

Structural equivalence Then we list several core properties associated to structural equivalence. The first one is the trivial observation that it is preserved by the action of permutations:

Proposition D.2. If $A \equiv B$ consist of n parallel subprocesses and $\pi \in S_n$, $\pi.A \equiv \pi.B$.

An important corollary is that the function $(\pi, A) \mapsto \pi.A$ is a group action on the set of extended processes quotiented by $\equiv$. In particular its stabilisers are permutation groups:

Corollary D.3. For all extended processes A consisting of n processes, $\mathrm{Stab}(A)$ is a subgroup of S_n.

An other property is that structural equivalence preserves static equivalence and skeletons. This is simply because static equivalence is preserved by bijective renaming of private names.

Proposition D.4. If $A \equiv B$ then $A \sim B$, and if $P \equiv_E Q$ then $\mathsf{skel}(P\wr) = \mathsf{skel}(Q\wr)$.

The goal of our reductions by symmetry is to identify subprocesses that will behave identically, so that the execution of one is prioritised over the others; we formalise this by a notion that identifies traces that are identical up to their labels and public channels. Formally we write $X \simeq_{Ch} Y$ when X and Y are two identical objets up to renaming of public channels, i.e. $X = Y\varrho$ for ϱ permutation of Ch_{pub}; then we say that two traces u, v are *almost identical*, written $u \cong v$, when there exists a session matching σ for u and v (notion introduced in Appendix A) and ϱ an α-renaming such that

$$u : A_0 \xrightarrow{[a_1]^{\ell_1}} \cdots \xrightarrow{[a_n]^{\ell_n}} A_n$$
$$v : (B_0 \xrightarrow{[b_1]^{\ell_1\sigma}} \cdots \xrightarrow{[b_n]^{\ell_n\sigma}} B_n)\varrho$$

where for all i, $a_i \simeq_{Ch} b_i$ and A_i, B_i are of the form

$$A_i = (\{\!\{[P_1]^{\ell'_1}, \ldots, [P_m]^{\ell'_m}\}\!\}, \Phi)$$
$$B_i \simeq_{Ch} (\{\!\{[Q_1]^{\ell'_1\sigma}, \ldots, [Q_m]^{\ell'_m\sigma}\}\!\}, \Phi')$$

with $P_j \equiv_E Q_j$ for all j and $\Phi =_E \Phi'$. We sometimes make the session matching explicit by writing $u \overset{\sigma}{\cong} v$. Almost identity is an equivalence relation on traces, which is mostly justified by Proposition A.3. We now identify some transformations that produce almost identical traces and we give the core property that we expect such traces to verify.

Proposition D.5. If π is consistent w.r.t. t then $t \overset{\pi}{\cong} t\pi$.

The proof of this proposition is straightforward ($\equiv_E$ and $\simeq_{Ch}$ can be replaced by syntactic equalities). An other property is:

Proposition D.6. If $u \overset{\pi}{\cong} v$ and ϱ is a permutation of Ch_{pub} then $u \overset{\pi}{\cong} v\varrho$. Besides if u and v have the same initial extended process B, $\pi_{|L} = \mathrm{id}$ for L the set of labels of B, and $t : A \overset{\mathrm{tr}}{\Longrightarrow} B$, then $t \cdot u \overset{\bar{\pi}}{\cong} t \cdot v$ where $\bar{\pi}$ is the extension of π that coincides with π on $dom(\pi)$ and is the identity on the labels of t.

But the most important property is that structurally equivalent processes have almost identical traces (again with the identity renaming of channels).

Proposition D.7. Let A_0, B_0 two extended processes such that $A_0 \equiv B_0$, and L the set of labels appearing in A_0 and B_0 (which is the same). Then for all traces $u \in \mathbb{T}(A_0)$, there exists $v \in \mathbb{T}(B_0)$ such that $u \overset{\pi}{\cong} v$ with $\pi_{|L} = \mathrm{id}$ and $u \sim v$.

Proof. Intuitively the trace v is constructed by mirroring all transitions of u in B_0. The technical part is that structural equivalence includes associative-commutative reordering of parallel operators, making the labels of u and v different after the first transition. Let us write $u : A_0 \xrightarrow{[a_1]^{\ell_1^u}} \cdots \xrightarrow{[a_n]^{\ell_n^u}} A_n$ and

$$A_0 = (\{\!\{[P_1]^{\ell_1^0}, \ldots, [P_m]^{\ell_m^0}\}\!\}, \Phi)$$
$$B_0 = (\{\!\{[Q_1]^{\ell_1^0}, \ldots, [Q_m]^{\ell_m^0}\}\!\}, \Phi)\varrho$$

for some α-renaming ϱ and $P_i \equiv_E Q_i$ for all i. We show by induction on n that there exists a trace

$$v : B_0 \xrightarrow{[a_1]^{\ell_1^u\pi}} \cdots \xrightarrow{[a_n]^{\ell_n^u\pi}} B_n$$

for some session matching π for u and v such that $\pi_{|L} = \mathrm{id}$, and A_i, B_i are of the form

$$A_i = (\{\!\{[R_1]^{\ell_1}, \ldots, [R_m]^{\ell_m}\}\!\}, \Phi_i)$$
$$B_i = (\{\!\{[S_1]^{\ell_1}, \ldots, [S_m]^{\ell_m}\}\!\}, \Phi'_i)\varrho$$

with $R_j \equiv_E S_j$ for all j and $\Phi_i =_E \Phi'_i$.

For $n = 0$ it suffices to choose v the empty trace and $\pi = \mathrm{id}$. For $n > 0$ we write $u = u_0 \cdot u'$ with $u_0 : A_0 \xrightarrow{\alpha} A_1$ and perform a case analysis on the rule from which the transition u_0 is derived. Below we write $L(A)$ the set of labels appearing in a process A.

▹ *case 1:* rules (In), (Out) or (Comm)

We write $v_0 : B_0 \xrightarrow{\alpha} B_1$, the transition obtained by performing the same action as u_0 in B_0 at the same label. In particular we apply the induction hypothesis to u' from A_1, B_1, which gives

$v' \in \mathbb{T}(B_1)$ and π a session matching for u' and v'. Then π is also a session matching for t and s and $\pi_{|L(A_0)} = \pi_{|L(A_1)} = \mathrm{id}$ and it suffices to choose $v = v_0 \cdot v'$.

▹ *case 2:* rule (Par)

Let us use the following notations

$$u_0 : A_0 = (\{\!\{ [\, \textstyle\prod_{i=1}^{p} P_i \,]^{\ell} \}\!\} \cup \{\!\{ [\, P_i \,]^{\ell_i} \}\!\}_{i=p+1}^{q}, \Phi)$$
$$\xrightarrow{\tau} (\{\!\{ [\, P_i \,]^{\ell \cdot i} \}\!\}_{i=1}^{p} \cup \{\!\{ [\, P_i \,]^{\ell_i} \}\!\}_{i=p+1}^{q}, \Phi)$$
$$B_0 = (\{\!\{ [\, Q \,]^{\ell} \}\!\} \cup \{\!\{ [\, Q_i \,]^{\ell_i} \}\!\}_{i=p+1}^{q}, \Phi)\varrho$$

with $\prod_{i=1}^{p} P_i \equiv_E Q$ and for all $i \in [\![p+1, n]\!]$, $P_i \equiv_E Q_i$. By Proposition D.4 and since Q is in $\rightsquigarrow$-normal form by definition, we have $Q = \prod_{i=1}^{p} Q_i$ where, for some permutation $\sigma \in S_p$, $Q_i \equiv_E P_{\sigma(i)}$ for all $i \in [\![1, p]\!]$. In particular by applying rule (Par) in B_0 at label ℓ we obtain a transition $v_0 : B_0 \xrightarrow{\tau} B_1$ such that we can apply the induction hypothesis from $A_1, B_1\pi_0$, where π_0 is the label permutation defined by $\pi_0(\ell \cdot i) = \ell \cdot \sigma(i)$ and $supp(\pi_0) \subseteq \{\ell \cdot 1, \ldots, \ell \cdot p\}$. Note that $B_1\pi_0$ is well defined since π_0 is consistent w.r.t. B_1. Let us thus apply the induction hypothesis to the processes $A_1, B_1\pi_0$ and the trace u'. We obtain a trace $v' \in \mathbb{T}(B_1\pi_0)$ and a session matching π' for u' and v'. To conclude it suffices to choose $v = v_0 \cdot v'\pi_0^{-1}$ and π defined by

$$\begin{array}{ll} \pi(\ell) = \ell & \\ \pi(\ell \cdot i \cdot \ell') = \pi_0^{-1}(\ell \cdot i) \cdot \ell'' & \text{where } \pi'(\ell \cdot i \cdot \ell') = \ell \cdot i \cdot \ell'' \\ \pi(\ell') = \pi'(\ell') & \text{if } \ell \text{ not a prefix of } \ell' \end{array}$$

π is well defined and a session matching for u and v because π' is a session matching for u' and v' such that $\pi'_{|L(A_1)} = \mathrm{id}$. □

Finally, almost identical traces preserve POR properties. Indeed if $u \overset{\sigma}{\cong} v$, a permutation π permutes independent blocks of u *iff* it permutes independent blocks of v and, in this case, $\pi.u \overset{\sigma}{\cong} \pi.v$. In addition, since the frames of almost identical traces are equal modulo theory and renaming of private names, a block is proper *iff* an almost identical block is. All in all:

Proposition D.8 . If $u \cong v$ then $u \in \mathbb{O}^{\vee}_{\mathsf{c+i^*}}$ *iff* $v \in \mathbb{O}^{\vee}_{\mathsf{c+i^*}}$.

D.2 Universal symmetries

D.2.1 Permutation groups

Let us first prove that the sets Sym_1 and Sym_2 (notation of Section 5.3.1) are subgroups of S_n. For that we recall the following classical characterisation of finite subgroups:

Proposition D.9 . Let G be a group and S a non-empty, finite subset of G that is closed under composition w.r.t. the group law of G. Then S is a subgroup of G.

Proof. It suffices to prove E is closed by inverse. Since E is closed under composition, if $x \in E$ then $x^n \in E$ for all $n > 1$. Besides E is also finite, hence there exists $i > j$ such that $x^i = x^j$. In particular by multiplying by x^{-j} (the inverse of x^j in G), we obtain that x^{i-j-1} is the inverse of x in E. □

In the next two theorems we fix the notations of Section 5.3.1 Sym_1 and Sym_2 depend of. We therefore consider a trace $t \in \mathbb{T}(P)$ whose final process A consists of n processes.

Proposition D.10 . Sym_1 is a subgroup of S_n.

Proof. Since $\mathsf{Stab}(A)$ is a group, it suffices to prove Sym_1^Q is a subgroup of S_n. We recall that it is the set of permutations $\pi \in S_n$ such that for all traces $t^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} A^2$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$, there exists a trace $s^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} \pi.A^2$ such that $t = \mathsf{fst}(s^2)$. This set contains the identity (it suffices to choose $s^2 = t^2$). Therefore by Proposition D.9 it suffices to prove that Sym_1^Q is closed by composition.

Let $\pi, \sigma \in \mathsf{Sym}_1^Q$. In particular there is $s^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} \pi.A^2$ such that $t = \mathsf{fst}(s^2)$. Since s^2 has the same final frames as t^2, we also have $\mathsf{fst}(s^2) \sim \mathsf{snd}(s^2)$. Therefore since $\pi' \in \mathsf{Sym}_1^Q$, there exists $u^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} \sigma.\pi.A^2 = (\sigma \circ \pi).A^2$ such that $\mathsf{fst}(u^2) = t$, hence $\pi \circ \sigma \in \mathsf{Sym}_1^Q$. □

Proposition D.11 . Sym_2 is a subgroup of S_n.

Proof. Rephrasing the definition, $\pi \in \mathsf{Sym}_2$ *iff* there exists ϱ permutation of Ch_{pub} such that $\pi.A \equiv A\varrho$ and, for all twin traces $t^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} A^2$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$, we have $\pi.\mathsf{snd}(A^2) \equiv \mathsf{snd}(A^2)\varrho$. Similarly to Sym_1, it suffices to prove that Sym_2 is closed by composition to conclude by Proposition D.9, since $\mathrm{id} \in \mathsf{Sym}_2$. Thus let $\pi, \pi' \in \mathsf{Sym}_2$ and ϱ, ϱ' permutations of Ch_{pub} such that

(1) $\pi.A \equiv A\varrho$ and $\pi'.A \equiv A'\varrho'$

(2) for all twin traces $t^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} A^2$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$, $\pi.\mathsf{snd}(A^2) \equiv \mathsf{snd}(A^2)\varrho$ and $\pi'.\mathsf{snd}(A^2) \equiv \mathsf{snd}(A^2)\varrho'$.

We prove that $\pi \circ \pi' \in \mathsf{Sym}_2$. First of all

$$(\pi \circ \pi').A \equiv \pi.(A\varrho') = (\pi.A)\varrho' \equiv A\varrho\varrho' .$$

Then let $t^2 : (P, Q) \overset{\mathsf{tr}}{\Longrightarrow} A^2$ such that $t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$. With the same reasoning as above we obtain $(\pi \circ \pi').\mathsf{snd}(A^2) \equiv \mathsf{snd}(A^2)\varrho\varrho'$, hence the conclusion. □

D.2.2 Correctness of the reduction by symmetry

First of all we prove that the optimisation consisting of discarding transitions based on the analysis of Sym. At first we only study the refinement of $\mathbb{O}^{\vee}_{\mathsf{c+i^*}}$ and tackle later the question of the compatibility with high-priority transitions and with the lexicographic reduction. We use again the notations of Section 5.3.1 and, if $\pi \in \mathsf{Sym}$ and $L = \{\ell_i\}_{i=1}^{n}$, we let $\bar{\pi}$ the label permutation defined by $supp(\bar{\pi}) \subseteq L$ and $\bar{\pi}(\ell_i) = \ell_{\pi(i)}$.

Proposition D.12 . Let $\pi \in \mathsf{Sym}$. We also let a trace $u \in \mathbb{O}^{\vee}_{\mathsf{c+i^*}}$ of the form

$$u = t \cdot (A \xrightarrow{[\, a \,]^{\ell}} B) \cdot v$$

We assume that for all traces $u' \cong u$ of the form

$$u' = t \cdot (A \xrightarrow{[\, a' \,]^{\ell\bar{\pi}}} B') \cdot v' \qquad (\star)$$

there exists $s \in \mathbb{T}(Q)$ with $u' \sim s$ and a session matching for u' and s. Then there exists $s \in \mathbb{T}(Q)$ with $u \sim s$ and a session matching for u and s.

Proof. We decompose the proof in three parts, isolating the characterisation based on Sym_1, the one based on Sym_2, and how to compose the two.

***Part 1**: Proof in the case $\pi \in \mathsf{Sym}_1$.* Let $u \in \mathbb{T}(P)$ a trace with the same notations as the statement of the theorem. We write

$$u_0 : A \xrightarrow{[\,a\,]^{\ell}} B$$

The permutation $\bar{\pi}$ is consistent w.r.t. A and the trace $(u_0 \cdot v)\bar{\pi}$ is therefore well defined, and of the form

$$(u_0 \cdot v)\bar{\pi} : (\pi^{-1}.A \xrightarrow{[\,a\,]^{\ell\bar{\pi}}} B) \cdot v\bar{\pi}$$

But $\pi \in \mathsf{Sym}_1$ and in particular $\pi \in \mathsf{Stab}(A)$. Since $\mathsf{Stab}(A)$ is a group we deduce, by definition, that $\pi^{-1}.A \equiv A$. Hence by Proposition D.7 there exists $w \in \mathbb{T}(A)$ such that $w \sim u_0 \cdot v$ and $(u_0 \cdot v)\bar{\pi} \overset{\sigma}{\cong} w$ for some σ such that $\sigma_{|L} = \mathrm{id}$. In particular $u \cong t \cdot w$ by Proposition D.6 and $t \cdot w$ is therefore of the form ($\star$). Hence by hypothesis there exists $s' \in \mathbb{T}(Q)$ such that $s' \sim t \cdot w$ and σ' a session matching for $t \cdot w$ and s'. We also let σ'_t and σ'_w the restrictions of σ' to the labels of t and w, respectively. If we decompose $s' = s'_t \cdot s'_w$ with $|s'_t| = |t|$ then σ'_t (resp. σ'_w) is a session matching for t and s'_t (resp. w and s'_w). In particular, $\sigma'_w \circ \sigma$ is a session matching for $(u_0 \cdot v)\bar{\pi}$ and s'_w. Thus $\phi = \sigma_w \circ \sigma \circ \bar{\pi}$ is a session matching for $u_0 \cdot v$ and s'_w.

Besides by hypothesis $\pi^{-1} \in \mathsf{Sym}_1^Q$ because Sym_1 is a group; in terms of session matchings this assumption can be rephrased as follows:

for all $z : Q \overset{\mathrm{tr}}{\Longrightarrow} Z$ and ω session matching for t and z, there exists $z' : Q \overset{\mathrm{tr}}{\Longrightarrow} Z$ and ω' session matching for t and z' such that $\omega'_{|L(Z)} = \omega_{|L(Z)} \circ \bar{\pi}$, where $L(Z)$ is the set of labels of Z.

In particular for $z = s'_t$ and $\omega = \sigma'_t$ we obtain $s''_t \in \mathbb{T}(Q)$ and σ''_t session matching for t and s''_t such that for all $i \in [\![1, n]\!]$,

$$\begin{aligned} \sigma''_t(\ell_i) &= \sigma'_t(\ell_{\pi(i)}) \\ &= \sigma'_w(\ell_{\pi(i)}) && \text{(because } \sigma'_t \text{ and } \sigma'_w \text{ coincide on } L) \\ &= \phi(\ell_i) && \text{(because } \sigma_{|L} = \mathrm{id}) \end{aligned}$$

This justifies that the following mapping ψ of domain $dom(\sigma''_t) \cup dom(\phi)$ is well defined

$$\psi_{|dom(\sigma''_t)} = \sigma''_t \qquad\qquad \psi_{|dom(\phi)} = \phi$$

All in all it sufficies to choose $s = s''_t \cdot s'_w$ since ψ is a session matching for u and s, and $s \sim u$ because

$$tr(s) = tr(s'_t) \cdot tr(w) = tr(t) \cdot tr(u_0 \cdot v) = tr(u) \qquad \square$$

***Part 2**: Proof in the case $\pi \in \mathsf{Sym}_2$.* We use the same initial notations for u and u_0 as in the previous Part, with in particular

$$(u_0 \cdot v)\bar{\pi} : (\pi^{-1}.A \xrightarrow{[\,a\,]^{\ell\bar{\pi}}} B) \cdot v\bar{\pi}$$

But by hypothesis $\pi \in \mathsf{Sym}_2$, or equivalently $\pi^{-1} \in \mathsf{Sym}_2$ since Sym_2 is a group. Therefore there exists ϱ permutation of Ch_{pub} such that $\pi^{-1}.A \equiv A\varrho$. Hence by Proposition D.7 there exists $w \in \mathbb{T}(A\varrho)$ such that $w \sim u_0 \cdot v$ and $u_0 \cdot v \overset{\sigma_a}{\cong} w$ for some σ_a such that $\sigma_{a|L} = \mathrm{id}$. In particular $u \cong t \cdot w\varrho^{-1}$ by Proposition D.6 and $t \cdot w\varrho^{-1}$ is therefore of the form ($\star$) (and its first action is $[\,a'\varrho^{-1}\,]^{\ell\bar{\pi}}$). Hence by hypothesis there exists $s' \in \mathbb{T}(Q)$ such that $s' \sim t \cdot w\varrho^{-1}$ and σ' a session matching for $t \cdot w\varrho^{-1}$ and s'. We also let σ'_t and σ'_w the restrictions of σ' to the labels of t and w, respectively. Decomposing $s' = s'_t \cdot s'_w$ with $|s'_t| = |t|$, σ'_t (resp. σ'_w) is a session matching for t and s'_t (resp. $w\varrho^{-1}$ and s'_w). Hence we have established so far:

- σ'_t is a session matching for t and s'_t
- $\sigma'_w \circ \sigma_a \circ \bar{\pi}$ is a session matching for $u_0 \cdot v$ and $s'_w\varrho$

However this is not sufficient to construct a session matching for $u = t \cdot u_0 \cdot v$, among other things because the last extended process of s'_t (let us write it Z) is not the same as the first extended process of $s'_w\varrho$ (which is $Z\varrho$). This is where we use the hypothesis $\pi^{-1} \in \mathsf{Sym}_2^Q$, which implies that $\pi^{-1} \in \mathsf{Stab}_{\varrho}(Z)$. In this instance of $\mathsf{Stab}_{\varrho}(Z)$, the implicit ordering of the labels of Z is the one mirroring the ordering on the labels of A: namely the labels of Z are ordered $\sigma'_w(\ell_1), \sigma'_w(\ell_2) \ldots, \sigma'_w(\ell_n)$. In particular in our context, the hypothesis $\pi^{-1} \in \mathsf{Stab}_{\varrho}(Z)$ means:

$$(\pi^{-1}.(Z\sigma'^{-1}_w))\sigma'_w \equiv Z\varrho$$

or more succinctly:

$$Z\phi \equiv Z\varrho^{-1} \qquad\qquad \text{with } \phi = \sigma'_w \circ \bar{\pi}^{-1} \circ \sigma'^{-1}_w$$

Hence by applying Proposition D.7 to the trace $s'_w\phi$, we obtain $s''_w \in \mathbb{T}(Z\varrho^{-1})$ such that $s''_w \sim s'_w$ and σ_z a session matching for $s'_w\phi$ and s''_w such that $\sigma_{z|L(Z)} = \mathrm{id}$ for $L(Z) = \{\sigma'_w(\ell) \mid \ell \in L\}$ the set of labels of Z. Therefore $\sigma_z \circ \phi$ is a session matching for $s'_w\varrho$ and $s''_w\varrho \in \mathbb{T}(Z)$. To conclude we choose $s = s'_t \cdot s''_w\varrho$. To define the session matching for u and s we consider

$$\psi_1 = \sigma'_t \qquad\qquad \begin{aligned} \psi_2 &= \sigma_z \circ \phi \circ \sigma'_w \circ \sigma_a \circ \bar{\pi} \\ &= \sigma_z \circ \sigma'_w \circ \bar{\pi}^{-1} \circ \sigma_a \circ \bar{\pi} \end{aligned}$$

We have established that ψ_1 is a session matching for t and s'_t, and that ψ_2 is a session matching for $u_0 \cdot v$ and s''_w. Therefore a session matching for u and s would be the matching ψ such that $dom(\psi) = dom(\psi_1) \cup dom(\psi_2)$ and $\psi_{dom(\psi_1)} = \psi_1$ and $\psi_{dom(\psi_2)} = \psi_2$. This function can be defined if ψ_1 and ψ_2 coincide on $dom(\psi_1) \cap dom(\psi_2) = L$. And indeed, for all $i \in [\![1, n]\!]$,

$$\begin{aligned} \psi_2(\ell_i) &= \sigma_z \circ \sigma'_w \circ \bar{\pi}^{-1} \circ \sigma_a(\ell_{\pi(i)}) && \text{(by definition)} \\ &= \sigma_z \circ \sigma'_w \circ \bar{\pi}^{-1}(\ell_{\pi(i)}) && \text{(because } \sigma_{a|L} = \mathrm{id}) \\ &= \sigma_z \circ \sigma'_w(\ell_i) \\ &= \sigma'_w(\ell_i) && \text{(because } \sigma_{z|L(Z)} = \mathrm{id}) \\ &= \psi_1(\ell_i) && \text{(because } \sigma_{t'|L} = \sigma_{w'|L}) \end{aligned}$$

Finally we also verify that $s \sim u$, in particular $tr(s) = tr(u)$:

$$tr(s) = tr(t) \cdot tr(s'_w\varrho) = tr(t) \cdot tr(w) = tr(t) \cdot tr(u_0 \cdot v) = tr(u) \quad \square$$

***Part 3**: General case.* That is, $\pi = \pi_1 \circ \cdots \circ \pi_n$ for some

$\pi_1, \dots, \pi_n \in \mathsf{Sym}_1 \cup \mathsf{Sym}_2$. For succinctness if $w \in \mathbb{T}(P)$ we write $H(w)$ the property

There exists $s \in \mathbb{T}(Q)$ such that $tr(s) = tr(w)$ and a session matching for w and s.

In particular the property we attempt to prove is that for all $\pi \in \mathsf{Sym}$, assuming that $H(u')$ for all $u' \cong u$ of the form $(\star)$, we have $H(u)$. We proceed by induction on a proof that $\pi \in \mathsf{Sym}$.

▹ *case 1: $\pi = \mathrm{id}$.*

Since Sym_1 is a group, it contains the identity and this case is already captured by the *Part 1* of the proof.

▹ *case 2: $\pi = \phi \circ \psi$ with $\phi \in \mathsf{Sym}_1 \cup \mathsf{Sym}_2$, $\psi \in \mathsf{Sym}$.*

To apply the induction hypothesis to ψ (and thus conclude the proof) we have to prove that $H(u'')$ for all traces $u'' \cong u$:

$$u'' : t \cdot (A \xrightarrow{[\,a''\,]^{\ell\bar{\psi}}} B'') \cdot v'' \,.$$

Let u'' be such a trace. Using either **Part 1** or **Part 2** (depending on whether $\phi \in \mathsf{Sym}_1$ or $\phi \in \mathsf{Sym}_2$) we know that it suffices to prove that $H(u')$ for all $u' \cong u''$ of the form

$$u' : t \cdot (A \xrightarrow{[\,a'\,]^{\ell\bar{\pi}}} B') \cdot v' \,.$$

Since $\cong$ is transitive all such u' also verify $u' \cong u$, hence the conclusion by hypothesis. □

This is the core technical lemma to justify the correctness of our universal symmetry.

D.2.3 Compatibility with POR

So far we have provided the core argument to prove that $\mathbb{O}^{\forall}_{\mathsf{sym}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c+i^*}}$. However it is more complex to prove it compatible with $\mathbb{O}^{\forall}_{\mathsf{por}}$, i.e. with high-priority transitions and the lexicographic reduction. In essence we will have to reuse the core corectness arguments developed in Appendices C.5 and C.6 to prove their correctness, but with the handicap of applying symmetries at the same time.

Proposition 5.1 . $\mathbb{O}^{\forall}_{\mathsf{por}} \cap \mathbb{O}^{\forall}_{\mathsf{sym}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{por}}$.

Proof. Since the reasoning will be the same anyway, we prove the stronger result that $\mathbb{O}^{\forall}_{\mathsf{sym}} \cap \mathbb{O}^{\forall}_{\mathsf{por}}$ is a correct refinement of $\mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$. We let $\sqsubseteq_{\mathsf{sym}}$ and $\sqsubseteq_0$ the notions of inclusion respectively induced by these two optimisations and show that they coincide. The inclusion $\sqsubseteq_0 \subseteq \sqsubseteq_{\mathsf{sym}}$ is immediate. Regarding the converse inclusion, let P, Q two processes such that $P \sqsubseteq_{\mathsf{sym}} Q$ and let us prove that $P \sqsubseteq_0 Q$. For that we prove that for all traces $u \in \mathbb{T}(P) \cap \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$, there exists $u^2 \in \mathbb{T}(P, Q)$ such that $u = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$. We proceed by well founded induction w.r.t. the ordering $\preccurlyeq_{lex}$ on compressed traces.

▹ *case 1: $u \notin \mathbb{O}^{\forall}_{\mathsf{por}}$*

We consider an arbitrary maximal extension $u' \in \mathbb{O}^{\forall}_{\mathsf{c}}$ obtained by executing arbitrary blocks after u as long as possible. Then by Propositions 4.9, 4.12 and 4.14 there exists $\bar{u} \equiv_{\mathsf{b\text{-}por}} u'$ such that $\bar{u} \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$. Since $\bar{u}$ is maximal, by Proposition 4.16 there exists π permuting independent blocks of $\bar{u}$ such that $\pi.\bar{u} \in \mathbb{O}^{\forall}_{\mathsf{por}} \subseteq \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ and $\pi.\bar{u} \prec_{lex} u$. In particular by applying the induction hypothesis to $\pi.\bar{u}$ we obtain u^2 such that $\pi.\bar{u} = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$. Hence the conclusion by Proposition C.3, since $\pi.\bar{u} \equiv_{\mathsf{b\text{-}por}} u$.

▹ *case 2: $u \in \mathbb{O}^{\forall}_{\mathsf{por}} \cap \mathbb{O}^{\forall}_{\mathsf{sym}}$*

Then the conclusion follows from the hypothesis $P \sqsubseteq_{\mathsf{sym}} Q$.

▹ *case 3: $u \in \mathbb{O}^{\forall}_{\mathsf{por}} \setminus \mathbb{O}^{\forall}_{\mathsf{sym}}$*

Let us write $u = u_1 \cdot b \cdot u_2$ with $u_1, u_2 \in \mathbb{O}^{\forall}_{\mathsf{c}}$ and $b \in \mathbb{T}(A)$ a block such that $b \notin \mathbb{O}^{\forall}_{\mathsf{sym}}$. Let $\ell_1, \dots, \ell_k$ be the labels A and ℓ the label of its first action.

▹ *case 3a:* the first action of b is an input.

We write $\ell = \ell_{i_0}$. Referring to the notations of the definition of $\mathbb{O}^{\forall}_{\mathsf{sym}}$ we let $\pi \in \mathsf{Sym}$ such that

$$\pi(i_0) = \min \mathrm{Orb}(i_0) \,.$$

The conclusion will follow from Proposition D.12 provided we manage to comply with its hypothesis. Thus let $u' \cong u$ a trace of the following form

$$u' : u_1 \cdot b' \cdot u_2'$$

where the first action of $b' \in \mathbb{T}(A)$ is an input on the label $\ell_{\pi(i_0)}$. We recall that $u' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ by Proposition D.8. Besides by hypothesis $u_1 \cdot b \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$, therefore no high-priority transitions are available at the start of b, i.e. in A (since the first transition of b is an input and can therefore not be high-priority). Therefore $u_1 \cdot b' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$. On the other hand there are no guarantees that high-priority transitions are respected in u_2'; however by Proposition 4.14 we can fix π permuting independent blocks of u_2' such that $\pi.u_2' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$. In particular if $u'' = u_1 \cdot b' \pi.u_2'$ we have $u'' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ but also and $u'' \prec_{lex} u$ because $tr(b') \prec tr(b)$ by definition of π. By the induction hypothesis applied to u'' we therefore obtain u^2 such that $u'' = \mathsf{fst}(u^2) \sim \mathsf{snd}(u^2)$. Using Proposition C.3 we then obtain v^2 such that $u' = \mathsf{fst}(v^2) \sim \mathsf{snd}(v^2)$. Hence the expected conclusion by Proposition A.1.

▹ *case 3b:* the first action of b is an internal communication.

We write $\ell = \ell_{i_0} \mid \ell_{i_1}$ where the label of the input is ℓ_{i_0}. Referring to the notations of the definition of $\mathbb{O}^{\forall}_{\mathsf{sym}}$ we let $\pi \in \mathsf{Sym}$ such that

$$(\pi(i_0), \pi(i_1)) = \min \mathrm{Orb}(i_0, i_1) \,.$$

Like the previous case we want to conclude by using Proposition D.12. Let $u' \cong u$ a trace of the following form

$$u' : u_1 \cdot b' \cdot u_2'$$

where the first action of b' is on the label $\ell_{\pi(i_0)} \mid \ell_{\pi(i_1)}$. We recall that by definition $\mathrm{Orb}(i_0, i_1) \subseteq IO$ and therefore that an internal communication at label $\ell_{\pi(i_0)} \mid \ell_{\pi(i_1)}$ respects high-priority transitions. In particular $u_1 \cdot b' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$ (using Proposition D.8 again). Besides, using a similar reasoning as in the case 3a we also obtain that $u' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*}}$ and a π permuting independent blocks of u_2' such that $u'' = u_1 \cdot b' \cdot \pi.u_2' \in \mathbb{O}^{\forall}_{\mathsf{c+i^*+0}}$. The conclusion of the reasoning is then similar, applying the induction hypothesis to u'' and then Propositions A.1 and C.3. □

D.3 Existential symmetries

In this section we prove the existential optimisation:

Proposition 5.3. $\mathbb{O}^{\exists}_{\mathsf{sym}}$ is a correct refinement of $\mathbb{O}^{\exists}_{\mathsf{all}}$.

However although the formalism of twin traces makes the formalisation of the optimisation more conceit as a simple restriction of rule (Match), we found it easier to prove it correct using the formalism of session matchings; that is, the optimisation rather consists of reducing the potential number of session matchings considered for a trace. Let us thus first state it in this paradigm.

Let $t \in \mathbb{T}(P)$ a trace and an instance of rule (Par) in t:

$$(\{\!\!\{ [\, \textstyle\prod_{i=1}^{n} P_i \,]^{\ell} \}\!\!\} \cup \mathcal{P}, \Phi) \xrightarrow{\tau} (\{\!\!\{ [\, P_i \,]^{\ell \cdot i} \}\!\!\} \cup \mathcal{P}, \Phi)$$

We also let $t' \in \mathbb{T}(Q)$ such that $tr(t) = tr(t')$ and assume that there exists a session matching σ for t and t'. In particular t' contains a unique application of rule (Par) of the form

$$(\{\!\!\{ [\, \textstyle\prod_{i=1}^{n} Q_i \,]^{\sigma(\ell)} \}\!\!\} \cup \mathcal{Q}, \Psi) \xrightarrow{\tau} (\{\!\!\{ [\, Q_i \,]^{\sigma(\ell)\cdot\pi(i)} \}\!\!\} \cup \mathcal{Q}, \Psi) = \pi^{-1}.A$$

for some $\pi \in S_n$ and $\sigma(\ell \cdot i) = \sigma(\ell) \cdot \pi(i)$. Then we write

$$\pi \sim \pi' \quad \textit{iff} \quad \exists u \in \mathsf{Stab}(A), \pi' = \pi \circ u$$

and say that σ is *well formed on ℓ* if π is minimal in its equivalence class for $\sim$ (w.r.t. an arbitrary fixed ordering on S_n). We say that σ is *well formed* when it is well formed on all labels ℓ on which an application of rule (Par) has been performed in t. In particular we obtain the following characterisation of $\mathbb{O}^{\exists}_{\mathsf{sym}}$ by Proposition A.1:

Proposition D.13. The following statements are equivalent for all processes P, Q

1. $\forall t \in \mathbb{T}(P), \exists t^2 \in \mathbb{T}(P,Q) \cap \mathbb{O}^{\exists}_{\mathsf{sym}}, t = \mathsf{fst}(t^2) \sim \mathsf{snd}(t^2)$
2. for all $t \in \mathbb{T}(P)$, there exists $t' \in \mathbb{T}(Q)$ and σ a well formed session matching for t and t' such that $t \sim t'$

In the remaining of the section we therefore focus on proving that the second item is equivalent to $P \sqsubseteq_s Q$. For that we rely again on the results of Appendix D.1.

Proposition D.14. The following statements are equivalent for all processes P, Q

1. $P \sqsubseteq_s Q$
2. for all $t \in \mathbb{T}(P)$, there exists $t' \in \mathbb{T}(Q)$ and σ a well formed session matching for t and t' such that $t \sim t'$

Proof. The implication (2) $\Rightarrow$ (1) is immediate. Conversely we assume (1), let $t \in \mathbb{T}(P)$ and thus $t' \in \mathbb{T}(Q)$ such that $t \sim t'$ and σ_0 a session matching for t and t'. We now construct $s \in \mathbb{T}(Q)$ and σ a well formed session matching for t and s. We proceed by (decreasing) induction on the number of transitions of t before the first (Par) transition on a label ℓ the matching σ_0 is not well formed on. If there are none σ_0 is well formed and it sufficies to choose $s = t'$ and $\sigma = \sigma_0$. Otherwise let ℓ be the first such label. We write $t = t_0 \cdot p \cdot t_1$ with p the transition

$$(\{\!\!\{ [\, \textstyle\prod_{i=1}^{n} P_i \,]^{\ell} \}\!\!\} \cup \mathcal{P}, \Phi) \xrightarrow{\tau} (\{\!\!\{ [\, P_i \,]^{\ell \cdot i} \}\!\!\} \cup \mathcal{P}, \Phi)\,.$$

Similarly we write $t' = t'_0 \cdot p' \cdot t'_1$ with p' the transition

$$(\{\!\!\{ [\, \textstyle\prod_{i=1}^{n} Q_i \,]^{\sigma_0(\ell)} \}\!\!\} \cup \mathcal{Q}, \Psi) \xrightarrow{\tau} (\{\!\!\{ [\, Q_i \,]^{\sigma_0(\ell)\cdot\pi(i)} \}\!\!\} \cup \mathcal{Q}, \Psi) = \pi^{-1}.A\,.$$

We let $u \in \mathsf{Stab}(A)$ such that $\pi_0 = \pi \circ u$ is minimal in its equivalence class for $\sim$. By definition of $\mathsf{Stab}(A)$ we have in particular $\pi^{-1}.A \equiv \pi_0^{-1}.A$. Therefore by Proposition D.7 we know that there exists $s_1 \in \mathbb{T}(\pi_0^{-1}.A)$ and a session matching σ_1 for t'_1 and s_1 such that $t'_1 \sim s_1$ and $\sigma_1(\ell') = \ell'$ for all labels $\ell' \in L$ the set of labels appearing in A. In particular we consider the trace

$$s' = t'_0 \cdot ((\{\!\!\{ [\, \textstyle\prod_{i=1}^{n} Q_i \,]^{\sigma_0(\ell)} \}\!\!\} \cup \mathcal{Q}, \Psi) \xrightarrow{\tau} \pi_0^{-1}.A) \cdot s_1$$

and σ' the label permutation defined by

$$\begin{aligned} \sigma'(\ell') &= \sigma_0(\ell') && \text{if } \ell' \text{ label of } t_0 \cdot p \\ \sigma'(\ell') &= \sigma_1 \circ \sigma_0(\ell') && \text{if } \ell' \text{ label of } t_1 \end{aligned}$$

Since σ_0 and σ_1 coincide on L, σ' is well defined a session matching for t and s'. Besides it is well formed on ℓ (and on all labels appearing before in the trace t). Hence we can apply the induction hypothesis to s' and σ' which gives the expected trace s and session matching σ. ◻