Dynamic Probabilistic Input Output Automata

PIERRE CIVIT, Sorbonne Université, CNRS, LIP6, France
MARIA POTOP-BUTUCARU, Sorbonne Université, CNRS, LIP6, France

We present probabilistic dynamic I/O automata, a framework to model dynamic probabilistic systems. Our work extends dynamic I/O Automata formalism of Attie & Lynch [1] to probabilistic setting. The original dynamic I/O Automata formalism included operators for parallel composition, action hiding, action renaming, automaton creation, and behavioral sub-typing by means of trace inclusion. They can model mobility by using signature modification. They are also hierarchical: a dynamically changing system of interacting automata is itself modeled as a single automaton. Our work extends to probabilistic settings all these features. Furthermore, we prove necessary and sufficient conditions to obtain the implementation monotonicity with respect to automata creation and destruction. Our construction uses a novel proof technique based on homomorphism that can be of independent interest. Our work lays down the foundations for extending composable secure-emulation of Canetti et al. [4] to dynamic settings, an important tool towards the formal verification of protocols combining probabilistic distributed systems and cryptography in dynamic settings (e.g. blockchains, secure distributed computation, cybersecure distributed protocols etc).

1 INTRODUCTION

Distributed computing area faces today important challenges coming from modern applications such as peer-to-peer networks, cooperative robotics, dynamic sensor networks, adhoc networks and more recently, cryptocurrencies and blockchains which have a tremendous impact in our society. These newly emerging fields of distributed systems are characterized by an extreme dynamism in terms of structure, content and load. Moreover, they have to offer strong guaranties over large scale networks which is usually impossible in deterministic settings. Therefore, most of these systems use probabilistic algorithms and randomized techniques in order to offer scalability features. However, the vulnerabilities of these systems may be exploited with the aim to provoke an unforeseen execution that diverges from the understanding or intuition of the developers. Therefore, formal validation and verification of these systems has to be realized before their industrial deployment.

The formalisation of distributed systems has been pioneered by Lynch and Tuttle [6]. They proposed the formalism of Input/Output Automata to model deterministic distributed system. Later, this formalism is extended by Segala in [8] with Markov decision processes [7]. In order to model randomized distributed systems Segala proposes Probabilistic Input/Output Automata. In this model each process in the system is an automaton with probabilistic transitions. The probabilistic protocol is the parallel composition of the automata modeling each participant.

The modelisation of dynamic behavior in distributed systems has been addressed by Attie & Lynch in [1] where they propose Dynamic Input Output Automata formalism. This formalism extends the Input/Output Automata with the ability to change their signature dynamically (i.e. the set of actions in which the automaton can participate) and to create other I/O automata or destroy existing I/O automata. The formalism introduced in [1] does not cover the case of probabilistic distributed systems and therefore cannot be used in the verification of recent blockchains such as Algorand [5].

In order to respond to the need of formalisation in secure distributed systems, Canetti & al. proposed in [2] task-structured probabilistic Input/Output automata (TPIOA) specifically designed for the analysis of cryptographic protocols. Task-structured probabilistic Input/Output automata are Probabilistic Input/Output automata extended with tasks that are equivalence classes on the set of actions. The task-structure allows a generalisation of “off-line scheduling” where the non-determinism of the system is resolved in advance by a task-scheduler, i.e. a sequence of tasks chosen in advance that trigger the actions among the enabled ones. They define the parallel composition for this type of automata. Inspired by
the literature in security area they also define the notion of implementation for TPQOA. Informally, the implementation of a Task-structured probabilistic Input/Output automata should look "similar" to the specification whatever will be the external environment of execution. Furthermore, they provide compositional results for the implementation relation. Even thought the formalism proposed in [2] has been already used in the verification of various cryptographic protocols this formalism does not capture the dynamicity of probabilistic dynamic systems such as peer-to-peer networks or blockchains systems where the set of participants dynamically changes.

**Our contribution.** In order to cope with dynamicity and probabilistic nature of modern distributed systems we propose an extension of the two formalisms introduced in [1] and [2]. Our extension uses a refined definition of probabilistic configuration automata in order to cope with dynamic actions. The main result of our formalism is as follows: the implementation of probabilistic configuration automata is monotonic to automata creation and destruction. That is, if systems $X_A$ and $X_B$ differ only in that $X_A$ dynamically creates and destroys automaton $A$ instead of creating and destroying automaton $B$ as $X_B$ does, and if $A$ implements $B$ (in the sense they cannot be distinguished by any external observer), then $X_A$ implements $X_B$. This result enables a design and refinement methodology based solely on the notion of externally visible behavior and permits the refinement of components and subsystems in isolation from the rest of the system. In our construction we exhibit the need of considering only creation-oblivious schedulers in the implementation relation, i.e. a scheduler that, upon the (dynamic) creation of a sub-automaton $A$, does not take into account the previous internal actions of $A$ to output (randomly) a transition. Surprisingly, the task-schedulers introduced by Canetti & al. [2] are not creation-oblivious. Interestingly, an important contribution of the paper of independent interest is the proof technique we used in order to obtain our results. Differently from [1] and [2] which build their constructions mainly on induction techniques, we developed an elegant homomorphism based technique which aim to render the proofs modular. This proof technique can be easily adapted in order to further extend our framework with cryptography and time.

It should be noted that our work is an intermediate step before extending composable secure-emulation [4] to dynamic settings. This extension is necessary for formal verification of secure dynamic distributed systems (e.g. blockchain systems).

**Paper organization.** The paper is organized as follow. Section 3 is dedicated to a brief introduction of the notion of probabilistic measure and recalls notations used in defining Signature I/O automata of [1]. Section 4 builds on the frameworks proposed in [1] and [2] in order to lay down the preliminaries of our formalism. More specifically, we introduce the definitions of probabilistic signed I/O automata and define their composition and implementation. In Section 5 we extend the definition of configuration automata proposed in [1] to probabilistic configuration automata then we define the composition of probabilistic configuration automata and prove its closeness. The key result of our formalisation, the monotonicity of PSIOA implementations with respect to creation and destruction, is presented in the end of Section 6 and demonstrated in the remaining sections, up to Section 11). Section 12 explains why the off-line scheduler introduced by Canetti & al. [4] is not creation-oblivious and therefore cannot be used to obtain our key result.

## 2 WARM UP

In this section we describe the paper in a very informal way. We aim to give some intuitions on the role of each section. The section 3 gives some preliminaries on probability and measure.
### 2.1 Probabilistic Signature Input/Output Automata (PSIOA)

The section 4 defines the notion of probabilistic signature Input/Output automata (PSIOA). A PSIOA $\mathcal{A}$ is an automaton that can move from one state to another through actions. The set of states of $\mathcal{A}$ is then denoted $\text{states}(\mathcal{A})$, while we note $\text{start}(\mathcal{A}) \in \text{states}(\mathcal{A})$ the unique start state of $\mathcal{A}$. At each state $q \in \text{states}(\mathcal{A})$ some actions can be triggered in its signature $\text{sig}(\mathcal{A})(q)$. Such an action leads to a new state with a certain probability. The measure of probability triggered by an action $a$ in a state $q$ is denoted $\eta_{\mathcal{A}(q,a)}$. The model aims to allow the composition (noted $\mathcal{A}_1||\cdots||\mathcal{A}_n$) of several automata to capture the idea of an interaction between them. That is why a signature is composed by three categories of actions: the input actions, the output actions and the internal actions. In practice the input actions of an automaton potentially aim to be the output action of another automaton and vice-versa. Hence an automaton can influence another one through a shared action. The comportment of the entire system is formalised by the automaton issued from the composition of the automata of the system.

![Diagram of two automata](image)

Fig. 1. A representation of two automata $U$ and $V$. In the top line, we see the PSIOA $U$ in a state $q_U^1$, s. t. $\text{sig}(U)(q_U^1) = (\text{out}(U)(q_U^1), \text{in}(U)(q_U^1), \text{int}(U)(q_U^1)) = (\{b,c\}, \{d\}, \{g\})$, the PSIOA $V$ in a state $q_V^1$, s. t. $\text{sig}(V)(q_V^1) = (\text{out}(V)(q_V^1), \text{in}(V)(q_V^1), \text{int}(V)(q_V^1)) = (\{d,e\}, \{c,f\}, \{h\})$ and the result of their composition, the PSIOA $U||V$ in a state $(q_U^1, q_V^1)$, s. t. $\text{sig}(U||V)(q_U^1, q_V^1) = (\text{out}(U||V)((q_U^1, q_V^1)), \text{in}(U||V)((q_U^1, q_V^1)), \text{int}(U||V)((q_U^1, q_V^1))) = (\{b,c,d,e\}, \{f\}, \{g,h\})$. In the second line we see the same PSIOA but in different states. We see the PSIOA $U$ in a state $q_U^2$, s. t. $\text{sig}(U)(q_U^2) = (\text{out}(U)(q_U^2), \text{in}(U)(q_U^2), \text{int}(U)(q_U^2)) = (\{b\}, \{a,j\}, \{g\})$, the PSIOA $V$ in a state $q_V^2$, s. t. $\text{sig}(V)(q_V^2) = (\text{out}(V)(q_V^2), \text{in}(V)(q_V^2), \text{int}(V)(q_V^2)) = (\{e,j\}, \{c\}, \{h,i\})$ and the result of their composition, the PSIOA $U||V$ in a state $(q_U^2, q_V^2)$, s. t. $\text{sig}(U||V)((q_U^2, q_V^2)) = (\text{out}(U||V)((q_U^2, q_V^2)), \text{in}(U||V)((q_U^2, q_V^2)), \text{int}(U||V)((q_U^2, q_V^2))) = (\{b,e,j\}, \{a,c\}, \{g,h,i\})$.

After this, we can speak about an execution of an automaton, which is an alternating sequence of states and actions. We can also speak about a trace of an automaton, which is the projection of an execution on the external actions uniquely. This allows us to speak about external behaviour of a system, that is, what we can observe from an outside point of view.

### 2.2 Scheduler

We remarked in the example of figure 2 that an inherent non-determinism has to be solved to be able to define a measure of probability on the executions, and so on the traces. This is the role of the scheduler which is a function
Fig. 2. The figure represents a tree of possible executions for a PSIOA $\mathcal{A}$. The red dots ($q_0, q_1, \ldots, q_3$) represent some states of the PSIOA. The PSIOA can move from one state to another through actions ($a, b, c, d, e, f, \ldots$) represented with colored solid arrows. Such an action $\text{act}$, triggered from a specific state $q$ does not lead directly to another state $q'$ but to a probabilistic distribution $\eta(A, q, \text{act})$ represented by a white dot and as many dashed black arrows as states in the support of $\eta(A, q, \text{act})$, i.e., the subset of states with non-zero probability for $\eta(A, q, \text{act})$. For example, the PSIOA $A$ can be in state $q_0$, trigger the action $a$ that leads him to $\eta(A, q, a)$ and hence to $q_1^v$ with probability $\frac{1}{4}$ and to $q_1^v$ with probability $\frac{3}{4}$. Some executions are $q_0, a, q_1^v, b, q_2^v, c, q_3^v, c, q_3^v$. Let’s assume that $b$ and $b'$ are internal actions. The set of traces is then $\{a; a, c; a, c'; d; d, e, d, f\}$. Typically $a, c = \text{trace}(q_0, a, q_1^v, b, q_2^v, c, q_3^v)$. We can already remark that a non-determinism is appearing since nothing states for the moment how to choose an action. How to know which action to take at state $q_0$ among $a$ and $d$. This non-determinism will be solved by the scheduler, introduced later.

$\sigma : Frags^*(\mathcal{A}) \rightarrow \text{SubDisc}(dtrans(\mathcal{A}))$ that (consistently) maps an execution fragment to a discrete sub-probability distributions on set of discrete transitions of the concerned PSIOA $\mathcal{A}$. Loosely speaking, the scheduler $\sigma$ decides (probabilistically) which transition to take after each finite execution fragment $\alpha$. Since this decision is a discrete sub-probability measure, it may be the case that $\sigma$ chooses to halt after $\alpha$ with non-zero probability: $1 - \sigma(\alpha)(dtrans(\mathcal{A})) > 0$.

A scheduler $\sigma$ and a probabilistic distribution $\mu$ on the set of finite execution fragments $Frags^*(\mathcal{A})$ generate a measure $\epsilon_{\sigma, \mu}$ on $\Sigma E\text{xec}_{\mathcal{A}}(\mathcal{A})$ generated by cones of execution fragments (of the form $C_{\alpha^x} = \{a^z \in Frags(\mathcal{A}) | a^z = \overline{a^x a^y} a^y \in Frags(\mathcal{A})\}$), and so a measure on the measurable space $(G, \mathcal{F}_G)$ for any measurable function $f$ from $(\mathcal{E}xec_{\mathcal{A}}(\mathcal{A}), \mathcal{F}E\text{xec}_{\mathcal{A}}(\mathcal{A}))$ to $(G, \mathcal{F}_G)$. Hence, when a scheduler is made explicit, we can state the probability that a cone of execution is reached and that a property holds. By default, the probabilistic distribution $\mu$ on the set of finite execution fragments $Frags^*(\mathcal{A})$ is $\delta_{\text{start}}(\mathcal{A})$, i.e., the Dirac distribution that has a measure of 1 for the start state of the concerned automaton $\mathcal{A}$. We denote by $\epsilon_{\sigma} = \epsilon_{\sigma, \delta_{\text{start}}(\mathcal{A})} : Execs(\mathcal{A}) \mapsto [0, 1]$ the execution distribution generated by the scheduler $\sigma$ and $\delta_{\text{start}}(\mathcal{A})$. 

4
2.3 Environment, external behavior, implementation

Now it is possible to define the crucial concept of implementation that captures the idea that an automaton Α "mimics" another automaton B. To do so, we define an environment E which takes on the role of a "distinguisher" for Α and B. In general, an environment of an automaton Α is just an automaton compatible with Α but some additional minor technical properties can be assumed. The set of environments of the automaton Α is denoted env(Α). The information used by an environment to attempt a distinction between two automata Α and B s. t. E ∈ env(Α) ∩ env(B) is captured by a function f_ε(Ε, Α) that we call insight function. In the literature, we very often deal with f_{ε(Ε, Α)} = trace_{ε(Ε, Α)}, but in our case, the theorem of implementation monotonicity (stated later) holds for a slightly different function that we call print(E, Α) which takes into account the entire execution of the environment itself. The philosophy of the approach remains nevertheless the same.

For any insight function f_ε(Ε, Α), we denote by f-dist_{Ε||Α}(σ) the image measure of ε_σ under f_{ε(Ε, Α)}. From here, this is classic to define the f-external behaviour of Α, denoted ExtBeh_{Ε||Α}^f : E ∈ env(Α) ↦→ {f-dist_{Α||Ε}(σ)|σ ∈ schedulers(Ε||Α)}. Such an object captures all the possible measures of probability on the external interaction of the concerned automaton Α and an arbitrary environment E. Finally, we can say that Α f-implements B if ∀E ∈ env(Α) ∩ env(Β), ExtBeh_{Ε||Α}^f(E) ⊆ ExtBeh_{Ε||Β}^f(E), i.e. for any 'distinguisher' E for Α and B, for any possible distribution f-dist_{Ε||Β}(σ') of the interaction between E and B generated by Α', i.e. for every external perception ζ ∈ range(f_{Ε||Β}(σ')) ⊆ range(f_{Ε||Β}(σ)), f-dist_{Ε||Β}(ζ) = f-dist_{Ε||Β}(σ').

However, as already mentioned in [8], the correctness of an algorithm may be based on some specific assumptions on the scheduling policy that is used. Thus, in general, we are interested only in a subset of schedulers(Ε||Α). A function that maps any automaton W to a subset of schedulers(Ε||Α) is called a scheduler schema. Among the most noteworthy examples are the fair schedulers, the off-line, a.k.a. oblivious schedulers, defined in opposition with the online-schedulers. So, we note ExtBeh_{Ε||Α}^{Sch} : E ∈ env(Α) ↦→ {f-dist_{Α||Ε}(σ)|σ ∈ Sch(Ε||Α)} where Sch is a scheduler schema and we say that Α f-implements B according to a scheduler schema Sch if ∀E ∈ env(Α) ∩ env(Β), ExtBeh_{Ε||Β}^{Sch}(E) ⊆ ExtBeh_{Ε||Β}^{Sch}(E). In the remaining, we will have a great interest for two certain classes of oblivious schedulers, i.e. i) the creation-oblivious scheduler (introduced later) and ii) the task-scheduler: an off-line scheduler already introduced in [2], which is relevant for cryptographic analysis. The previous notions can be adapted with a particular class of scheduler schema.

2.4 Probabilistic Configuration Automata (PCA)

The section 5 introduces the notion of probabilistic configuration automata (PCA). (see figure 4). A PCA is very closed to a PSIOA, but each state is mapped to a configuration C = (Α, S) which is a pair constituted by a set Α of PSIOA and the current states of each member of the set (with a mapping function S : Α → q_Α ∈ states(Α)). The idea is that the composition of the attached set can change during the execution of a PCA, which allows us to formalise the notion of dynamism, that is the potential creation and potential destruction of a PSIOA in a dynamic system. Some particular precautions have to be taken to make it consistent.
Fig. 3. An environment $E$, which is nothing more than a PSIOA compatible with both $A$ and $B$, tries to distinguish $A$ from $B$. We say that $A$ implements $B$ if for every environment $E$, $E$ is unable to distinguish $A$ from $B$. To formalise it a little bit more, but not totally, we say that $A$ implements $B$ if for every environment $E \in \text{env}(A) \cap \text{env}(B)$, for every scheduler $\sigma \in \text{scheduler}(E||A)$ applied to $E||A$ it exists a scheduler $\sigma' \in \text{scheduler}(E||B)$ applied to $E||B$ s. t. $\text{tdist}_{E||A}(\sigma) = \text{tdist}_{E||B}(\sigma')$, i.e. for every trace $\beta \in \text{trace}(E||A) \cup \text{trace}(E||B)$, $\text{tdist}_{E||A}(\sigma)(\beta) = \text{tdist}_{E||B}(\sigma')(\beta)$.

2.5 Road to monotonicity

The rest of the paper is dedicated to the proof of implementation monotonicity. We show that, under certain technical conditions, automaton creation is monotonic with respect to external behavior inclusion, i.e. if a system $X$ creates automaton $A$ instead of (previously) creating automaton $B$ and the external behaviors of $A$ are a subset of the external behaviors of $B$, then the set of external behaviors of the overall system is possibly reduced, but not increased. Such an external behavior inclusion result enable a design and refinement methodology based solely on the notion of externally visible behavior, and which is therefore independent of specific methods of establishing external behavior inclusion. It permits the refinement of components and subsystems in isolation from the entire system. To do so, we develop different mathematical tools.

2.5.1 Execution-matching. First, we define the notion of executions-matching (see figure 5) to capture the idea that two automata have the same comportment along some corresponding executions. Basically an execution-matching from a PSIOA $A$ to a PSIOA $B$ is a morphism $f^{EX}: \text{Execs}'_{A} \rightarrow \text{Execs}(B)$ where $\text{Execs}'_{A} \subseteq \text{Execs}(A)$. This morphism preserves some properties along the pair of matched executions: signature, transition, ... in such a way that for every pair $(\alpha, \alpha') \in \text{Execs}(A) \times \text{Execs}(B)$ s. t. $\alpha' = f^{EX}(\alpha)$, $\epsilon_{\sigma}(\alpha) = \epsilon_{\sigma'}(\alpha')$ for every pair of scheduler $(\sigma, \sigma')$ (so-called alter ego) that are "very similar" in the sense they take into account only the "structure" of the argument to return a sub-probability distribution, i.e. $\alpha' = f^{EX}(\alpha)$ implies $\sigma(\alpha) = \sigma'(\alpha')$. When the executions-matching is a bijection function from $\text{Execs}(A)$ to $\text{Execs}(B)$, we say $A$ and $B$ are semantically-equivalent (they differ only syntactically).
Dynamic Probabilistic Input Output Automata

Fig. 4. The figure represents an execution fragment \((q^1_X, c, q^2_X, h, q^3_X, b, q^4_X)\) of a PCA \(X\). In the left column, we see different states \(q^1_X, q^2_X, q^3_X\) and \(q^4_X\) of the PCA \(X\), represented with white diamonds (\(\Diamond\)). Each of these states \(q^i_X\) is mapped through the mapping \(\text{config}(X)\) (represented with right dotted arrows) to a configuration \(C^i_X\), represented with a white triangle (\(\triangle\)). For example, the state \(q^1_X\) is mapped with the configuration \(C^1_X = (A^1, S^1)\) with \(A^1 = \{U, V\}\), \(S^1(U) = q^1_U\) and \(S^1(V) = q^1_V\). The signature of the PCA \(X\) at state \(q^i_X\) is the one of the composition of automata, in their current states in the attached configuration \(C^i_X\); that are hidden and become internal for \(X\). For example, the configuration \(C^1_X\) has a signature \(\text{sig}(C^1_X) = (\text{out}(C^1_X), \text{int}(C^1_X)) = (\{b, e, c, d\}, \{a, f\})\), while the signature of \(X\) at corresponding state is \(\text{sig}(X)(q^1_X) = (\text{out}(X)(q^1_X), \text{int}(X)(q^1_X)) = (\{b, e, c\}, \{a, f\})\) since the unique action \(d\) in \(\text{hidden-actions}(X)(q^1_X)\) is hidden and hence becomes an internal action. We can define discrete transitions for configurations in a similar way as what we do for PSIOA, but adding some tools (formally defined in section 5) to allow the creation and the destruction of automata. For example, the automaton \(V\) is destroyed during the step \((q^1_X, h, q^1_X)\), while \(W\) is created during the step \((q^2_X, b, q^3_X)\) which is made explicit by the fact that \(\text{created}(X)(q^2_X)(b) = \{X\}\) where \(\text{created}(X)\) is a mapping function defined for any PCA. Some intuitive consistency rules have to be respected by pair of ‘corresponding transitions’ \(((q^1_X, \text{act}, \eta_X(q^1_X, \text{act})), (C^1_X, \text{act}, \eta_X(C^1_X, \text{Q})))\) represented by pair of parallel downward arrows (one from two diamonds \(\circ\) and one from two triangles \(\triangleright\)). For example, the probability \(\eta_X(q^1_X, \text{act}) (q^1_X)\) of reaching \(q^1_X\) by triggering \(c\) from \(q^1_X\) is equal to the probability \(\eta_X(C^1_X, q^1_X, \text{act}) (C^1_X)\) of reaching \(C^1_X\) by triggering \(c\) from \(C^1_X\). Moreover, a configuration transition has to respect some of other consistency rules with respect to the sub-automata that compose the configuration. Typically, the destruction of \(V\) in step \((C^2_X, h, C^3_X)\) comes from the fact that the action \(h\) triggered from state \(q^1_V\) of sub-automaton \(V\) lead to a probabilistic states distribution \(\eta_V(q^1_V, h)\) equal to \(\delta_{q^1_V}\) which is a Dirac distribution for a special state \(q^1_V\) with \(\text{sig}(V)(q^1_V) = (0, 0, 0)\) that means \(V\) "has been destroyed".
Pierre Civit and Maria Potop-Butucaru

Fig. 5. The figure represents the respective executions tree of two automata \( A \) and \( B \) with some strong similarities. The states of \( A \) (resp. \( B \)) are represented with red (resp. blue) dots. The actions are represented with solid arrows. An action leads to a discrete probability distribution on states \( \eta \), represented with a white dot and dashed arrows reaching the different states of the support of \( \eta \). In section 7, we define these strong similarities with what we call an executions-matching \((f, f^{\text{tr}}, f^{\text{ex}})\) where \( f : \text{states}_{A} \rightarrow \text{states}(B) \), \( f^{\text{tr}} : \text{dtrans}_{A} \rightarrow \text{dtrans}(B) \), \( f^{\text{ex}} : \text{Execs}_{A} \rightarrow \text{Execs}(B) \) with \( \text{states}_{A} \subseteq \text{states}(A) \), \( \text{dtrans}_{A} \subseteq \text{dtrans}(A) \), \( \text{Execs}_{A} \subseteq \text{Execs}(A) \). The mappings \( f, f^{\text{tr}} \) and \( f^{\text{ex}} \) preserves the important properties: signature for corresponding states, name of the action and measure of probability of corresponding states for corresponding transitions, etc. In the example the similarities exist until the states \( q_6, q_8 \) and \( q_9 \), hence we have \( \text{states}_{A} = \{ q_0, q_1, \ldots, q_9 \} \). The states-matching \( f \) is then defined s. t. \( \forall k \in [1, 9], f(q_k) = \tilde{q}_k \). Thereafter, we define define \( \text{Act} \) and \( f^{\text{tr}} \), s. t. \( \forall k \in [1, 9], \forall \text{act} \in \text{Act}, \forall \text{transition} \) \((q_k, \text{act}, \eta_{\text{A},q_k,\text{act}})\) \( \rightarrow \) \((\tilde{q}_k, \text{act}, \eta_{\text{B},\tilde{q}_k,\text{act}})\). Each pair of mapped transition gives the same probability to pair of mapped states, e. g. \( \eta_{\text{A},q_k^d,\text{act}}(q_k^d) = \eta_{\text{B},\tilde{q}_k^d,\text{act}}(\tilde{q}_k^d) \). Then we can define \( \text{Execs}_{A} \subseteq \text{Execs}(A) \) the set of executions composed only with states in \( Q_{\text{A}} \) and actions in \( \text{Act} \). Finally \( f^{\text{ex}} : \alpha = q_0^a \ldots a^n f(q_n) \in \text{Execs}_A \rightarrow f(q_0^a \ldots a^n f(q_n)) \) is an execution-matching. The Point is that if two schedulers \( \sigma \) and \( \sigma' \) only look at the preserved properties to output a measure of probability on the actions to take, the attached measures of probability will be equal, i. e. \( \epsilon_{\sigma}(\alpha) = \epsilon_{\sigma'}(\alpha') \).

2.5.2 A PCA \( X_A \) deprived from a PSIOA \( A \). Second, we define in section 8 the notion of a PCA \( X_A \) deprived from a PSIOA \( A \) noted \((X_A \setminus \{ A \})\). Such an automaton corresponds to the intuition of a similar automaton where \( A \) is systematically removed from the configuration of the original PCA (see figure 6 and 7).

2.5.3 Reconstruction: \((X_A \setminus \{ A \})||\tilde{A}^{\text{sw}}\). Thereafter we show in section 9 that under technical minor assumptions \( X_A \setminus \{ A \} \) and \( \tilde{A}^{\text{sw}} \) are composable where \( \tilde{A}^{\text{sw}} \) and \( A \) are semantically equivalent in the sense loosely introduced in the section 2.5.1. In fact \( \tilde{A}^{\text{sw}} \) is the simpleton wrapper of \( A \), that is a PCA that only owns \( A \) in its attached configuration (see figure 8). Let us note that if \( A \) implements \( B \), then \( \tilde{A}^{\text{sw}} \) implements \( B^{\text{sw}} \).
Dynamic Probabilistic Input Output Automata

Fig. 6. Projection on PCA, part 1/2: The figure represents a PCA $X$ like in figure 4. A sub-automaton $T$ (in purple) appears in the configurations attached to the states visited by $X$. The PCA $Y = X \setminus \{T\}$ where the sub-automaton $T$ is systematically removed is represented in figure 7.

Then we show that there is an (incomplete) execution-matching from $X_A$ to $(X_A \setminus \{A\})||\tilde{A}_{sw}$ (see figure 9). The domain of this executions-matching is the set of executions where $A$ is not (re-)created.

After this, we always try to reduce any reasoning on $X_A$ (resp. $X_B$) on a reasoning on $(X_A \setminus \{A\})||\tilde{A}_{sw}$ (resp. $(X_B \setminus \{B\})||\tilde{B}_{sw}$).

2.5.4 Corresponding PCA. We show in section 10 that under certain reasonable technical assumptions (captured in the definition of corresponding PCA w. r. t. $\mathcal{A}$, $\mathcal{B}$) that $(X_A \setminus \{A\})$ and $(X_B \setminus \{B\})$ are semantically-equivalent. We can note $Y$ an arbitrary PCA semantically-equivalent to $(X_A \setminus \{A\})$ and $(X_B \setminus \{B\})$. Finally, a reasoning on $E||X_A$ (resp. $E||X_B$) can be reduced to a reasoning on $(E||Y)||\tilde{A}_{sw} = E'||\tilde{A}_{sw}$ (resp. $E'||\tilde{B}_{sw}$) with $E' = E||Y$. Since $\tilde{A}_{sw}$ implements $\tilde{B}_{sw}$, we have already some results on $E'||\tilde{A}_{sw}$ and $E'||\tilde{B}_{sw}$ and so on $E||X_A$ and $E||X_B$ but only in a subset of executions (as long as neither $A$ nor $B$ is (re-)created, their comportment are the same). This reduction is represented in the figure 10.

Cut-paste execution fragments with $\mathcal{A}$ (resp. $\mathcal{B}$) creation at the endpoints. The reduction roughly described in figure 10 holds only for executions fragments that do not create the automata $\mathcal{A}$ and $\mathcal{B}$ after their destruction (or at very last action). Some technical precautions have to be taken to be allowed to paste these fragments together to finally say that
Fig. 7. Projection on PCA, part 2/2: the figure represents the PCA $Y = X \setminus \{T\}$ while the original PCA $X$ is represented in figure 6. We can see that the sub-automaton $T$ (in purple in figure 6) has been systematically removed from the configurations attached to the states visited by $Y$.

$A$ implements $B$ implies $X_A$ implements $X_B$. In fact, such a pasting is generally not possible for a fully information online scheduler. This observation motivated us to introduce the creation-oblivious scheduler that outputs (randomly) a transition without taking into account the internal actions of a sub-automaton $A$ preceding its last destruction. Surprisingly, the fully-offline task-scheduler introduced in [2] (slightly modified to be adapted to dynamic setting) is not creation-oblivious and so does not verify monotonicity.

3 PRELIMINARIES ON PROBABILITY AND MEASURE

We assume our reader is comfortable with basic notions of probability theory, such as $\sigma$-fields and (discrete) probability measures. A measurable space is denoted by $(S, F_S)$, where $S$ is a set and $F_S$ is a $\sigma$-algebra over $S$ that is $F_S \subseteq P(S)$, is closed under countable union and complementation and its members are called measurable sets (denotes the power set of $S$). A measure over $(S, F_S)$ is a function $\eta : F_S \rightarrow \mathbb{R}_{\geq 0}$, such that $\eta(\emptyset) = 0$ and for every countable collection of disjoint sets $\{S_i\}_{i \in I}$ in $F_S$, $\eta(\bigcup_{i \in I} S_i) = \Sigma_{i \in I} \eta(S_i)$. A probability measure (resp. sub-probability measure) over $(S, F_S)$ is a measure $\eta$ such that $\eta(S) = 1$ (resp. $\eta(S) \leq 1$). A measure space is denoted by $(S, F_S, \eta)$ where $\eta$ is a measure on $(S, F_S)$. 
The product measure space \((S_1, \mathcal{F}_{S_1}, \eta_1) \otimes (S_2, \mathcal{F}_{S_2}, \eta_2)\) is the measure space \((S_1 \times S_2, \mathcal{F}_{S_1} \otimes \mathcal{F}_{S_2}, \eta_1 \otimes \eta_2)\), where \(\mathcal{F}_{S_1} \otimes \mathcal{F}_{S_2}\) is the smallest \(\sigma\)-algebra generated by sets of the form \(A \times B\) \(A \in \mathcal{F}_{S_1}, B \in \mathcal{F}_{S_2}\) and \(\eta_1 \otimes \eta_2\) is the unique measure s. t. for every \(C_1 \in \mathcal{F}_{S_1}, C_2 \in \mathcal{F}_{S_2}, \eta_1 \otimes \eta_2(C_1 \times C_2) = \eta_1(C_1) \cdot \eta_2(C_2)\). If \(S\) is countable, we note \(\mathcal{P}(S) = 2^S\). If \(S_1\) and \(S_2\) are countable, we have \(2^{S_1} \otimes 2^{S_2} = 2^{S_1 \times S_2}\).

A discrete probability measure on a set \(S\) is a probability measure \(\eta\) on \((S, 2^S)\), such that, for each \(C \subset S, \eta(C) = \sum_{c \in C} \eta(c)\). We define \(\text{Disc}(S)\) and \(\text{SubDisc}(S)\) to be respectively, the set of discrete probability and sub-probability measures on \(S\). In the sequel, we often omit the set notation when we denote the measure of a singleton set. For a discrete probability measure \(\eta\) on a set \(S\), \(\text{supp}(\eta)\) denotes the support of \(\eta\), that is, the set of elements \(s \in X\) such that \(\eta(s) \neq 0\). Given set \(S\) and a subset \(C \subset S\), the Dirac measure \(\delta_c\) is the discrete probability measure on \(S\) that assigns probability 1 to \(C\). For each element \(s \in S\), we note \(\delta_s\) for \(\delta_{\{s\}}\).

If \(\{\mu_i\}_{i \in I}\) is a countable family of measures on \((S, \mathcal{F}_S)\), and \(\{p_i\}_{i \in I}\) is a family of non-negative values, then the expression \(\sum_{i \in I} p_i \mu_i\) denotes a measure \(m\) on \((S, \mathcal{F}_S)\) such that, for each \(C \in \mathcal{F}_S, m(C) = \sum_{i \in I} p_i \mu_i(C)\). A function \(f : X \rightarrow Y\) is said to be measurable from \((X, \mathcal{F}_X) \rightarrow (Y, \mathcal{F}_Y)\) if the inverse image of each element of \(\mathcal{F}_Y\) is an element of \(\mathcal{F}_X\).
Fig. 9. The figure shows the similarities between two PCA $X$ and $Z = (X \setminus \{V\}) || V^{sw}$ represented in the top line. The two components of $Z$, i.e. $(X \setminus \{V\})$ and $V^{sw}$ are represented in the bottom line like in figure 7 and 8. These similarities are captured by the notions of executions-matching and hold as long as the the sub-automaton $V$ is not created after a destruction. The idea is to reduce any reasoning on $X$ to reasoning on $(X \setminus \{V\}) || V^{sw}$.

of $\mathcal{F}_X$, that is, for each $C \in \mathcal{F}_Y$, $f^{-1}(C) \in \mathcal{F}_X$. In such a case, given a measure $\eta$ on $(X, \mathcal{F}_X)$, the function $f(\eta)$ defined on $\mathcal{F}_Y$ by $f(\eta)(C) = \eta(f^{-1}(C))$ for each $C \in Y$ is a measure on $(Y, \mathcal{F}_Y)$ and is called the image measure of $\eta$ under $f$. 

12
Dynamic Probabilistic Input Output Automata

Fig. 10. The figure represents successive steps to reduce the problem of an environment $E$ that tries to distinguish two PCA $X_A$ and $X_B$ (represented at first column) to a problem of an environment $E_D$ that tries to distinguish the automata $A$ and $B$ (represented at last column). The second column just remarks that the only difference between $X_A$ and $X_B$ is that $A$ supplants $B$ in $X_A$. The third column consist in the steps of deprivation (see section 2.5.2) and reconstruction (see section 2.5.3). The fourth column rearrange the parenthesis by associativity of the parallel composition to highlight $E_A = E_\parallel (X_A \{A\})$ and $E_B = E_\parallel (X_B \{B\})$ as respective environments of $A^{sw}$ and $B^{sw}$. In last column, we remark that $E_A$ and $E_B$ are semantically equivalent so there it is like we deal with a common environment $E_C$ for both $A^{sw}$ and $B^{sw}$. If we consider only the psioa components, the problem is reduced to a common environment $E_D$ that tries to distinguish the automata $A$ and $B$. The reasoning holds only as long as the automata $A$ and $B$ are not created after their destructions.

4 PROBABILISTIC SIGNATURE INPUT/OUTPUT AUTOMATA (PSIOA)

This section aims to introduce the first brick of our formalism, i.e. the probabilistic signature input/output automata (PSIOA). A PSIOA is the result of the generalization of probabilistic input/output automata (PIOA) [8] and signature input/output automata (SIOA) [1]. A PSIOA is thus an automaton that can randomly move from one state to another in response to some actions. The set of possible actions is the signature of the automaton and is partitioned into input, output and internal actions. An action can often be both the input of one automaton and the output of another one to captures the idea that the behavior of an automaton can influence the behavior of another one. As for the SIOA [1], the signature of a PSIOA can change according to the current state of the automaton, which allows us to formalise dynamicity later. The figure 11 gives a first intuition of what is a PSIOA.

4.1 Action Signature

We use the signature approach from [1].
Fig. 11. A representation of two automata $U$ and $V$. In the top line, we see the PSIOA $U$ in a state $q_U^1$, s. t. $\text{sig}(U)(q_U^1) = (\text{out}(U)(q_U^1),\text{in}(U)(q_U^1),\text{int}(U)(q_U^1)) = (\{b,c\},\{d\},\{q\})$, the PSIOA $V$ in a state $q_V^1$, s. t. $\text{sig}(V)(q_V^1) = (\text{out}(V)(q_V^1),\text{in}(V)(q_V^1),\text{int}(V)(q_V^1)) = (\{d,e\},\{c,f\},\{h\})$ and the result of their composition, the PSIOA $U||V$ in a state $(q_U^1,q_V^1)$, s. t. $\text{sig}(U||V)(q_U^1,q_V^1) = (\text{out}(U||V)(q_U^1,q_V^1),\text{in}(U||V)(q_U^1,q_V^1),\text{int}(U||V)(q_U^1,q_V^1)) = (\{b,c,d,e\},\{f\},\{g,h\})$. In the second line we see the same PSIOA but in different states. We see the PSIOA $U$ in a state $q_U^2$, s. t. $\text{sig}(U)(q_U^2) = (\text{out}(U)(q_U^2),\text{in}(U)(q_U^2),\text{int}(U)(q_U^2)) = (\{b\},\{a\},\{q\})$, the PSIOA $V$ in a state $q_V^2$, s. t. $\text{sig}(V)(q_V^2) = (\text{out}(V)(q_V^2),\text{in}(V)(q_V^2),\text{int}(V)(q_V^2)) = (\{e\},\{c\},\{h\})$ and the result of their composition, the PSIOA $U||V$ in a state $(q_U^2,q_V^2)$, s. t. $\text{sig}(U||V)(q_U^2,q_V^2) = (\text{out}(U||V)(q_U^2,q_V^2),\text{in}(U||V)(q_U^2,q_V^2),\text{int}(U||V)(q_U^2,q_V^2)) = (\{b,e\},\{a,c\},\{g,h,l\})$.

We assume the existence of a countable set $\text{Autids}$ of unique probabilistic signature input/output automata (PSIOA) identifiers, an underlying universal set $\text{Auts}$ of PSIOA, and a mapping $\text{aut} : \text{Autids} \rightarrow \text{Auts}$. $\text{aut}(A)$ is the PSIOA with identifier $A$. We use "the automaton $A$" to mean "the PSIOA with identifier $A$". We use the letters $A, B$, possibly subscripted or primed, for PSIOA identifiers. The executable actions of a PSIOA $A$ are drawn from a signature $\text{sig}(A)(q) = (\text{in}(A)(q),\text{out}(A)(q),\text{int}(A)(q))$, called the state signature, which is a function of the current state $q$ of $A$.

\[ \text{in}(A)(q), \text{out}(A)(q), \text{int}(A)(q) \]

are pairwise disjoint sets of input, output, and internal actions, respectively. We define $\text{ext}(A)(q)$, the external signature of $A$ in state $q$, to be $\text{ext}(A)(q) = (\text{in}(A)(q),\text{out}(A)(q))$.

We define $\text{local}(A)(q)$, the local signature of $A$ in state $q$, to be $\text{local}(A)(q) = (\text{out}(A)(q),\text{int}(A)(q))$. For any signature component, generally, the $\cup$ operator yields the union of sets of actions within the signature, e.g., $\text{sig}(A) : q \in Q \Rightarrow \text{sig}(A)(q) = \text{in}(A)(q) \cup \text{out}(A)(q) \cup \text{int}(A)(q)$. Also we define $\text{acts}(A) = \bigcup_{q \in Q} \text{sig}(A)(q)$, that is $\text{acts}(A)$ is the "universal" set of all actions that $A$ could possibly trigger, in any state. In the same way $\text{UI}(A) = \bigcup_{q \in Q} \text{in}(A)(q)$, $\text{UO}(A) = \bigcup_{q \in Q} \text{out}(A)(q)$, $\text{UH}(A) = \bigcup_{q \in Q} \text{int}(A)(q)$, $\text{UL}(A) = \bigcup_{q \in Q} \text{local}(A)(q)$, $\text{UE}(A) = \bigcup_{q \in Q} \text{ext}(A)(q)$.

4.2 PSIOA

We combine the SIOA of [1] with the PION of [8]:

**Definition 4.1 (PSIOA).** A PSIOA $A = (Q_A, \tilde{Q}_A, \text{sig}(A), D_A)$, where:

- $Q_A$ (a.k.a. $\text{states}(A)$) is a countable set of states, $(Q_A, Z_{Q_A})$ is a measurable space called the state space,
- $\tilde{Q}_A$ (a.k.a. $\text{start}(A)$) is the unique start state.
Dynamic Probabilistic Input Output Automata

- \( \text{sig}(A) : q \in Q_A \mapsto \text{sig}(A)(q) = (\text{in}(A)(q), \text{out}(A)(q), \text{int}(A)(q)) \) is the signature function that maps each state to a triplet of mutually disjoint countable set of actions, respectively called input, output and internal actions.

- \( D_A \subseteq Q_A \times \text{acts}(A) \times \text{Disc}(Q_A) \) (\( D_A \) a. k. a. \( \text{dtrans}(A) \)) is the set of probabilistic discrete transitions where \( \forall(q, a, \eta) \in D_A : a \in \text{sig}(A)(q) \). If \( (q, a, \eta) \) is an element of \( D_A \), we write \( q \xrightarrow{a} \eta \) and action \( a \) is said to be enabled at \( q \).

In addition \( A \) must satisfy the following conditions:

- \( E_1 \) (action enabling): \( \forall q \in Q_A : \forall a \in \text{sig}(A)(q), \exists \eta \in \text{Disc}(Q_A) : (q, a, \eta) \in D_A. \)

- \( T_1 \) (Transition determinism): For every \( q \in Q_A \) and \( a \in \text{sig}(A)(q) \) there is at most one \( \eta(A,q,a) \in \text{Disc}(Q_A) \), such that \( (q,a,\eta(A,q,a)) \in D_A. \)

**Notation.** For every \( \text{PSIOA} \ A \triangleq (Q_A, \bar{q}_A, \text{sig}(A), D_A) \), we note \( \text{states}(A) \triangleq Q_A, \text{start}(A) \triangleq \bar{q}_A, \text{dtrans}(A) \triangleq D_A. \) We also note \( \text{steps}(A) \triangleq \{(q, a, q') \in Q_A \times \text{acts}(A) \times Q_A|\exists (q, a, \eta) \in D_A, q' \in \text{supp}(\eta)\} \).

### 4.3 Execution, Trace

We use the classic notions of execution and trace from [8] to speak about the comportment of a PSIOA.

**Definition 4.2** (fragment, execution and trace of PSIOA). An execution fragment of a PSIOA \( A = (Q_A, \bar{q}_A, \text{sig}(A), D_A) \) is a finite or infinite sequence \( \alpha = q^0a^1q^1a^2... \) of alternating states and actions, such that:

1. If \( \alpha \) is finite, it ends with a state.
2. For any non-final state \( q_i \), \( (q_i, a^i, q^{i+1}) \in \text{steps}(A) \)

We write \( fstate(\alpha) \) for \( q^0 \) (the first state of \( \alpha \)), and if \( \alpha \) is finite, we write \( lstate(\alpha) \) for its last state. We note \( \text{states}(\alpha) \) (resp. \( \text{actions}(\alpha) \)) for the set of states (resp. actions) that compose \( \alpha \). The length \( |\alpha| \) of a finite execution fragment \( \alpha \) is the number of transitions along \( \alpha \). The length of an infinite execution fragment \( \alpha \) is infinite, \( (|\alpha| = \omega) \). If we use \( \text{Frags}(A) \) (resp. \( \text{Frags}^*(A) \)) to denote the set of all (resp. all finite) execution fragments of \( A \). An execution of \( A \) is an execution fragment \( \alpha \) with \( fstate(\alpha) = \bar{q} \). \( \text{Execs}(A) \) (resp. \( \text{Execs}^*(A) \)) denotes the set of all (resp. all finite) executions of \( A \). The trace of an execution fragment \( \alpha \), written \( \text{trace}(\alpha) \), is the restriction of \( \alpha \) to the external actions of \( A \). We say that \( \beta \) is a trace of \( A \) if there is \( \alpha \in \text{Execs}(A) \) with \( \beta = \text{trace}(\alpha) \). \( \text{Traces}(A) \) (resp. \( \text{Traces}^*(A) \)) denotes the set of all (resp. all finite) traces of \( A \). We define a concatenation operator \( \sim \) for execution fragments as follows. If \( \alpha = q^0a^1q^1a^2... \in \text{Frags}(A) \) and \( \alpha' = q'^0b'^1s^1... \in \text{Frags}(A) \), we define \( \alpha \sim \alpha' = q^0a^1q'^1a^2b'^1s^1... \) only if \( s^0 = q^0 \), otherwise \( \alpha \sim \alpha' \) is undefined. Hence the notation \( \alpha \sim s^0b'^1s^1... \) implicitly means \( s^0 = lstate(\alpha) \). We also note \( \alpha \sim (b^1, s^1) \) to states \( \alpha \sim lstate(\alpha)b'^1s^1 \). Let \( a, a' \in \text{Frags}(A) \), then \( a \) is a proper prefix of \( a' \) iff \( \exists \alpha'' \in \text{Frags}(A) \) such that \( a' = \alpha''a' \). In that case, we note \( a < a' \). We note \( a \leq a' \) if \( a < a' \) or \( a = a' \) and say that \( a \) is a prefix of \( a' \). We also overload \( \sim \) and use it for concatenating traces in the obvious manner.

**Definition 4.3** (reachable state). Let \( A = (Q_A, \bar{q}_A, \text{sig}(A), D_A) \) be a PSIOA. A state \( q \in Q_A \) is said reachable if it exists a finite execution that ends with \( q \). We note \( \text{reachables}(A) \) the set of reachable states of \( A \). We also note for every \( s \in \mathbb{N} \), \( \text{reachable}_{\leq s}(A) = \{q^* \in \text{reachables}(A)|\exists z \leq s, \exists q = q^0a^1q^1q^2...q^{z-1}a^zq^*\} \). This is the set of states reachable in

---

1 The conjunction of conditions \( E_1 \) and \( T_1 \) could allow us to model \( D_A \) as a partial function from \( Q_A \times \text{acts}(A) \) to \( \text{Disc}(Q_A) \). However, we keep this presentation to stay as close as possible to the usual notation of the literature. For the same reasons, we use both \( A \triangleq (Q_A, \bar{q}_A, \text{sig}(A), D_A) \) and \( \mathcal{A} \triangleq (\text{states}(A), \text{start}(A), \text{sig}(A), \text{dtrans}(A)) \).
less than \( s \) actions. We also note for every \( s \in \mathbb{N}^* \), \( \text{reachable}_s(\mathcal{A}) = \text{reachable}_{\leq s}(\mathcal{A}) \setminus \text{reachable}_{s-1}(\mathcal{A}) \). This is the set of states reachable in \( s \) actions but not less. By convention, \( \text{reachable}_0(\mathcal{A}) = \{ \hat{q}, \mathcal{A} \} \).

The set of sets \( \{ \text{reachable}_s(\mathcal{A}) \setminus \\mathbb{N}^* \} \) is clearly a partition of \( \text{reachable}(\mathcal{A}) \).

### 4.4 Compatibility and composition

The main aim of IO formalism is to compose several automata \( \mathcal{A} = \{ \mathcal{A}_1, \ldots, \mathcal{A}_n \} \) and obtain some guarantees of the system by composition of the guarantees of the different elements of the system. Some syntactic rules have to be satisfied before defining the composition operation.

**Definition 4.4 (Compatible signatures).** Let \( S \) be a set of signatures. Then \( S \) is compatible iff, \( \forall \text{sig}, \text{sig}' \in S, \text{ where sig} = (\text{in}, \text{out}, \text{int}), \text{sig}' = (\text{in}', \text{out}', \text{int}') \) and \( \text{sig} \neq \text{sig}' \), we have: 1. \( (\text{in} \cup \text{out} \cup \text{int}) \cap \text{int}' = \emptyset \), and 2. \( \text{out} \cap \text{out}' = \emptyset \).

**Definition 4.5 (Composition of Signatures).** Let \( \Sigma = (\text{in}, \text{out}, \text{int}) \) and \( \Sigma' = (\text{in}', \text{out}', \text{int}') \) be compatible signatures. Then we define their composition \( \Sigma \times \Sigma' = (\text{in} \cup \text{in}' - (\text{out} \cup \text{out}'), \text{out} \cup \text{out}', \text{int} \cup \text{int}')^2 \).

Signature composition is clearly commutative and associative.

Now we can define the compatibility of several automata with the compatibility of their attached signatures. First we define compatibility at a state, and discrete transition for a set of automata for a particular compatible state.

**Definition 4.6 (partially compatible at a state).** Let \( \mathcal{A} = \{ \mathcal{A}_1, \ldots, \mathcal{A}_n \} \) be a set of PSIOA. A state of \( \mathcal{A} \) is an element \( q = (q_1, \ldots, q_n) \in Q_\mathcal{A} = Q\mathcal{A}_1 \times \ldots \times Q\mathcal{A}_n \). We say \( \mathcal{A}_1, \ldots, \mathcal{A}_n \) are partially-compatible at state \( q \) (or \( \mathcal{A} \) is partially-compatible at state \( q \)) if \( \{ \text{sig}(\mathcal{A}_1)(q_1), \ldots, \text{sig}(\mathcal{A}_n)(q_n) \} \) is a set of compatible signatures. In this case we note \( \text{sig}(\mathcal{A})(q) = \text{sig}(\mathcal{A}_1)(q_1) \times \ldots \times \text{sig}(\mathcal{A}_n)(q_n) \) as per definition 4.5 and we note \( \eta_{(\mathcal{A},q,a)} \in \text{Disc}(Q_\mathcal{A}) \) s.t. for every action \( a \in \text{sig}(\mathcal{A})(q) \), \( \eta_{(\mathcal{A},q,a)} = \eta_{1} \otimes \ldots \otimes \eta_{n} \in \text{Disc}(Q_\mathcal{A}) \) that verifies for every \( j \in [1,n] \):

- If \( a \in \text{sig}(\mathcal{A}_j)(q_j), \eta_j = \eta_{(\mathcal{A}_j, q_j, a)} \).
- Otherwise, \( \eta_j = \delta_{q_j} \) (where \( \delta_{q_j} \) the Dirac distribution with \( \text{supp}(\delta_{q_j}) = \{ q_j \} \) and \( \delta_{q_j}(q_j) = 1 \))

which means \( \eta_{(\mathcal{A},q,a)} = \delta_{q} \) if \( a \not\in \text{sig}(\mathcal{A})(q) \).

We will say a set of automata is partially-compatible, if any reachable state is compatible. This motivates the two following definitions.

**Definition 4.7 (pseudo execution).** Let \( \mathcal{A} = \{ \mathcal{A}_1, \ldots, \mathcal{A}_n \} \) be a set of PSIOA. A pseudo execution fragment of \( \mathcal{A} \) is a finite or infinite sequence \( \alpha = q^0 a^1 q^1 a^2 \ldots \) of alternating states of \( \mathcal{A} \) and actions, such that:

- If \( \alpha \) is finite, it ends with a state of \( \mathcal{A} \).
- For every non final state \( q^j \), \( \mathcal{A} \) is partially-compatible at \( q^j \).
- For every action \( a^i, a'^i \in \text{sig}(\mathcal{A})(q^{i-1}) \).
- For every state \( q^i \), with \( i > 0, q^i \in \text{supp}(\eta_{(\mathcal{A}, q^i-1, a^i)}) \).

A pseudo execution of \( \mathcal{A} \) is a pseudo execution fragment of \( \mathcal{A} \) with \( q^0 = (\hat{q}, \mathcal{A}_1, \ldots, \hat{q}, \mathcal{A}_n) \).

**Definition 4.8 (reachable state).** Let \( \mathcal{A} = \{ \mathcal{A}_1, \ldots, \mathcal{A}_n \} \) be a set of PSIOA. A state \( q \) of \( \mathcal{A} \) is reachable if it exists a pseudo execution \( \alpha \) of \( \mathcal{A} \) ending on state \( q \).

Now we are able to define compatibility for a set of PSIOA.

\(^{2}\)not to be confused with Cartesian product. We keep this notation to stay as close as possible to the literature.
Definition 4.9 (partially-compatible PSIOA). Let $A = \{A_1, \ldots, A_n\}$ be a set of PSIOA. The automata $A_1, \ldots, A_n$ are $\ell$-partially-compatible with $\ell \in \mathbb{N}$, if no pseudo-execution $\alpha$ of $A$ with $|\alpha| \leq \ell$ ends on a non-partially-compatible state $q$. The automata $A_1, \ldots, A_n$ are partially-compatible if $A$ is partially-compatible at each reachable state $q$, i.e. if $A$ is $\ell$-partially-compatible for every $\ell \in \mathbb{N}$.

Finally, we can formally define our operation of composition. This is the central operation of any IOA formalism.

Definition 4.10 (partially-compatible PSIOA composition). If $A = \{A_1, \ldots, A_n\}$ is a partially-compatible set of PSIOA, with $A_i = (Q_{A_i}, \bar{q}_{A_i}, \sigma(A_i), D_{A_i})$, then their partial-composition $A_1 || \cdots || A_n$, is defined to be $A = (Q_A, \bar{q}_A, \sigma(A), D_A)$, where:

- $Q_A = \{q \in Q_{A_1} \times \cdots \times Q_{A_n} | q$ is a reachable state of $A\}$.
- $\bar{q}_A = (\bar{q}_{A_1}, \ldots, \bar{q}_{A_n})$
- $\sigma(A) : q = (q_1, \ldots, q_n) \in Q_A \mapsto \sigma(A)(q) = \sigma(A_1)(q_1) \times \cdots \times \sigma(A_n)(q_n)$ as per definition 4.5.
- $D_A \subset Q_A \times \text{acts}(A) \times \text{Disc}(Q_A)$ is the set of triples $(q, a, \eta_{(A,q,a)})$ so that $q \in Q_A$ and $a \in \text{sig}(A)(q)$.

This formalism extends the one proposed in [1] where it is required that all (potentially non-reachable) states are compatible. In addition to being slightly less restrictive, this notion of composability i) may facilitate the expression of mobile agents moving from one system to another (see section 5) and ii) will allow the proof of theorem of implementation monotonicity w.r.t. PSIOA creation (see section 6).
Given a parallel composition \( \mathcal{A} = \mathcal{A}_1 || ... || \mathcal{A}_n \) of \( n \) PSIOA, we define the projection of an execution fragment of \( \mathcal{A} \) onto one of the \( \mathcal{A}_i \), \( i \in [1 : n] \), in the usual way: the state components for all PSIOA other than \( \mathcal{A}_i \) are removed, and so are all actions in which \( \mathcal{A}_i \) does not participate.

**Definition 4.11 (Execution projection for PSIOA).** Let \( \mathcal{A} = \mathcal{A}_1 || ... || \mathcal{A}_n \) be a PSIOA. Let \( \alpha = q^0 a^1 ... a^n q^n \in \text{Frags}(\mathcal{A}) \). Then, \( \forall i \in [1 : n] \), we define \( \alpha \upharpoonright \mathcal{A}_i \) to be the sequence resulting from:

1. replacing each \( q^j = (q_i^j, ..., q_n^j) \) by its \( i \)-th component \( q_i^j \) and then
2. removing all \( a^j q_i^j \) s.t. \( a^j \notin \text{sig}(\mathcal{A}_i) (q_i^{j-1}) \).

The idea behind execution projection is to retain only the state of \( \mathcal{A}_i \), and only the actions which \( \mathcal{A}_i \) participates in. It has been shown in [1] (theorem 4, page 11), that for every PSIOA \( \mathcal{A} = \mathcal{A}_1 || ... || \mathcal{A}_n \), \( \forall \alpha \in \text{Execs}(\mathcal{A}) \), \( \forall i \in [1 : n] \), \( \alpha \downharpoonright \mathcal{A}_i \in \text{Execs}(\mathcal{A}_i) \).

### 4.5 Scheduler: define a measure on executions and traces

An inherent non-determinism appears for composable input/output (I/O) automata. Indeed, after composition (or even before), it is natural to obtain a state with several enabled actions. The most common case is the reception of two concurrent messages in flight from two different processes. This non-determinism must be solved if we want to define a probability measure on the automata executions and be able to say that a situation is likely to occur or not. To solve the non-determinism, we use a scheduler that chooses an enabled action from a signature.

#### 4.5.1 Scheduler: general definition

A scheduler is hence a function that takes an execution fragment as input and outputs the probability distribution on the set of transitions that will be triggered. We reuse the formalism from [8] with the syntax from [2].

**Definition 4.12 (scheduler).** A scheduler of a PSIOA \( \mathcal{A} \) is a function

\[ \sigma : \text{Frags}^*(\mathcal{A}) \rightarrow \text{SubDisc}(\text{dtrans}(\mathcal{A})) \]

such that \( (q, a, \eta) \in \text{supp}(\sigma(\alpha)) \) implies \( q = \text{Istate}(\alpha) \). Here \( \text{SubDisc}(\text{dtrans}(\mathcal{A})) \) is the set of discrete sub-probability distributions on \( \text{dtrans}(\mathcal{A}) \). Loosely speaking, \( \sigma \) decides (probabilistically) which transition to take after each finite execution fragment \( \alpha \). Since this decision is a discrete sub-probability measure, it may be the case that \( \sigma \) chooses to halt after \( \alpha \) with non-zero probability: \( 1 - \sigma(\alpha) (\text{dtrans}(\alpha)) > 0 \). We note \( \text{schedulers}(\mathcal{A}) \) the set of schedulers of \( \mathcal{A} \).

**Definition 4.13 (measure \( \epsilon_{\sigma,\alpha} \) generated by a scheduler and a fragment).** A scheduler \( \sigma \) and a finite execution fragment \( \alpha \) generate a measure \( \epsilon_{\sigma,\alpha} \) on the sigma-field \( \mathcal{F}_{\text{Execs}}(\mathcal{A}) \) generated by cones of execution fragments, where each cone \( C_{\alpha'} \) is the set of execution fragments that have \( \alpha' \) as a prefix, i.e. \( C_{\alpha'} = \{ \alpha \in \text{Execs}(\mathcal{A}) | \alpha' \leq \alpha \} \)

The measure of a cone \( C_{\alpha'} \) is defined recursively as follows:

\[
\epsilon_{\sigma,\alpha}(C_{\alpha'}) = \begin{cases} 
0 & \text{if both } \alpha' \nleq \alpha \text{ and } \alpha \nleq \alpha' \\
1 & \text{if } \alpha' \leq \alpha \\
\epsilon_{\sigma,\alpha}(C_{\alpha''}) \cdot \sigma(\alpha'') (\eta(\mathcal{A}_{\alpha',\alpha}) \cdot \eta(\mathcal{A}_{\alpha'',\alpha}) (q)) & \text{if } \alpha \leq \alpha' \text{ and } \alpha' = \alpha'' \text{ and } q' \sim q' a q
\end{cases}
\]

Standard measure theoretic arguments ensure that \( \epsilon_{\sigma,\alpha} \) is well-defined. We call the state \( \text{fstate}(\alpha) \) the first state of \( \epsilon_{\sigma,\alpha} \) and denote it by \( f\text{state}(\epsilon_{\sigma,\alpha}) \). If \( \alpha \) consists of the start state \( \text{start}(\mathcal{A}) \) only, we call \( \epsilon_{\sigma,\alpha} \) a probabilistic execution of \( \mathcal{A} \). Let \( \mu \) be a discrete probability measure over \( \text{Frags}^*(\mathcal{A}) \). We denote by \( \epsilon_{\sigma,\mu} \) the measure \( \sum_{\alpha \in \text{supp}(\mu)} \mu(\alpha) \cdot \epsilon_{\sigma,\alpha} \) and we say that \( \epsilon_{\sigma,\mu} \) is generated by \( \sigma \) and \( \mu \). We call the measure \( \epsilon_{\sigma,\mu} \) a generalized probabilistic execution fragment of \( \mathcal{A} \). If every execution fragment in \( \text{supp}(\mu) \) consists of a single state, then we call \( \epsilon_{\sigma,\mu} \) a probabilistic execution fragment of \( \mathcal{A} \).
Fig. 13. Non-deterministic execution: The scheduler allows us to solve the non-determinism, by triggering an action among the enabled one. Typically after execution $\alpha = q^0 d q^1 x$, the actions $e$ and $f$ are enabled and the probability to take one transition is given by the scheduler $\sigma$ that computes $\sigma(\alpha)$.

4.5.2 Scheduler Schema. Without restriction, a scheduler could become a too powerful adversary for practical applications. Hence, it is common to only consider a subset of schedulers, called a scheduler schema. Typically, a classic limitation is often described by a scheduler with "partial online information". Some formalism has already been proposed in [8] (section 5.6) to impose the scheduler that its choices are correlated for executions fragments in the same equivalence class where both the equivalence relation and the correlation must to be defined. This idea has been reused and simplified in [3] that defines equivalence classes on actions, called tasks. Then, a task-scheduler (a.k.a. "off-line" scheduler) selects a sequence of tasks $T_1, T_2, ...$ in advance that it cannot modify during the execution of the automaton. After each transition, the next task $T_i$ triggers an enabled action if there is no ambiguity and is ignored otherwise. One of our main contribution, the theorem of implementation monotonicity w.r.t. PSIOA creation, is ensured only for a certain scheduler schema, so-called creation-oblivious. However, we will see that the practical set of task-schedulers are not creation-oblivious.

Definition 4.14 (scheduler schema). A scheduler schema is a function that maps any PSIOA $W$ to a subset of $\text{.schedulers}(W)$.

4.6 Implementation

In last subsection, we defined a measure of probability on executions with the help of a scheduler to solve non-determinism. Now we can define the notion of implementation. The intuition behind this notion is the fact that any environment $E$ that would interact with both $A$ and $B$, would not be able to distinguish $A$ from $B$. The classic use-case is to formally show that a (potentially very sophisticated) algorithm implements a specification.
For us, an environment is simply a partially-compatible automaton, but in practice, he will play the role of a "distinguisher".

**Definition 4.15 (Environment).** A probabilistic environment for PSIOA \( \mathcal{A} \) is a PSIOA \( \mathcal{E} \) such that \( \mathcal{A} \) and \( \mathcal{E} \) are partially-compatible. We note \( \text{env}(\mathcal{A}) \) the set of environments of \( \mathcal{A} \).

Now we define insight function which is a function that captures the insights that could be obtained by an external observer to attempt a distinction.

**Definition 4.16 (Insight function).** An insight-function is a function \( f(\_\_\_\_\_) \) parametrized by a pair \((\mathcal{E}, \mathcal{A})\) of PSIOA where \( \mathcal{E} \in \text{env}(\mathcal{A}) \) so that for every PSIOA \( \mathcal{E} \), it exists a measurable space \((G_\mathcal{E}, \mathcal{F}_{G_\mathcal{E}})\), s. t. for every pair \((\mathcal{A}, \mathcal{B})\) of PSIOA where \( \mathcal{E} \in \text{env}(\mathcal{A}) \cap \text{env}(\mathcal{B}) \), \( f(\mathcal{E}, \mathcal{A}) \) (resp. \( f(\mathcal{E}, \mathcal{B}) \)) is a measurable function from \((\text{Execs}(\mathcal{E}||\mathcal{A}), \mathcal{F}_{\text{Execs}(\mathcal{E}||\mathcal{A})})\) (resp. \((\text{Execs}(\mathcal{E}||\mathcal{B}), \mathcal{F}_{\text{Execs}(\mathcal{E}||\mathcal{B})})\)) to \((G_\mathcal{E}, \mathcal{F}_{G_\mathcal{E}})\).

The point is that the arrival space \((G_\mathcal{E}, \mathcal{F}_{G_\mathcal{E}})\) is the same for the two functions \( f(\mathcal{E}, \mathcal{A}) \) and \( f(\mathcal{E}, \mathcal{B}) \) to enable a comparison. Some examples of insight-functions are the trace function and the print function introduced later.

Since an insight-function \( f(\_\_\_\_\_\_) \) is measurable, we can define the image measure of \( \epsilon_{\sigma, \mu} \) under \( f(\mathcal{E}, \mathcal{A}) \), i. e. the probability to obtain a certain external perception under a certain scheduler \( \sigma \) and a certain probability distribution \( \mu \) on the starting executions.

**Definition 4.17 \( (f\text{-dist}) \).** Let \( f(\_\_\_\_\_) \) be an insight-function. Let \( (\mathcal{E}, \mathcal{A}) \) be a pair of PSIOA where \( \mathcal{E} \in \text{env}(\mathcal{A}) \). Let \( \mu \) be a probability measure on \((\text{Execs}(\mathcal{E}||\mathcal{A}), \mathcal{F}_{\text{Execs}(\mathcal{E}||\mathcal{A})})\), and \( \sigma \in \text{scheduler}(\mathcal{E}||\mathcal{A}) \). We define \( f\text{-dist}(\mathcal{E}, \mathcal{A})(\sigma, \mu) \), to be the image measure of \( \epsilon_{\sigma, \mu} \) under \( f(\mathcal{E}, \mathcal{A}) \) (i. e. the function that maps any \( C \in \mathcal{F}_{G_\mathcal{E}} \) to \( \epsilon_{\sigma, \mu}(f(\mathcal{E}, \mathcal{A})(C)) \)). We note \( f\text{-dist}(\mathcal{E}, \mathcal{A})(\sigma, \delta_{\text{start}}(\mathcal{E}, \mathcal{A})) \).

We can see next definition of \( f\) implementation as the incapacity of an environment to distinguish two automata if it uses only information filtered by the insight function \( f\).

**Definition 4.18 \( (f\text{-implementation}) \).** Let \( f(\_\_\_\_\_\_) \) be an insight-function. Let \( \text{Sch} \) be a scheduler schema. We say that \( \mathcal{A} \) \( f\text{-implements} \) \( \mathcal{B} \) according to \( \text{Sch} \), noted \( \mathcal{A} \leq_{\text{Sch}}^f \mathcal{B} \), if \( \forall \mathcal{E} \in \text{env}(\mathcal{A}) \cap \text{env}(\mathcal{B}) \), \( \forall \sigma \in \text{Sch}(\mathcal{E}||\mathcal{A}) \), \( \exists \sigma' \in \text{Sch}(\mathcal{E}||\mathcal{B}) \), \( f\text{-dist}(\mathcal{E}, \mathcal{A})(\sigma) = f\text{-dist}(\mathcal{E}, \mathcal{B})(\sigma') \).

We states a necessary and sufficient condition to obtain composability of \( f\)-implementation.

**Definition 4.19.** Let \( f(\_\_\_\_\_\_) \) be an insight-function. We say that \( f(\_\_\_\_\_\_) \) is stable by composition if for every quadruplet of PSIOA \((\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}, \mathcal{E})\), s. t. \( \mathcal{B} \) is partially compatible with \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \), \( \mathcal{E} \in \text{env}(\mathcal{B}||\mathcal{A}_1) \cap \text{env}(\mathcal{B}||\mathcal{A}_2) \), for every \((C_1, C_2) \in \mathcal{F}_{\text{Execs}(\mathcal{E}||\mathcal{B})||\mathcal{A}_1} \times \mathcal{F}_{\text{Execs}(\mathcal{E}||\mathcal{B})||\mathcal{A}_2} \), \( f(\mathcal{E}, \mathcal{B}||\mathcal{A}_1)(C_1) = f(\mathcal{E}, \mathcal{B}||\mathcal{A}_2)(C_2) \Rightarrow f(\mathcal{E}, \mathcal{B}||\mathcal{A}_1)(C_1) = f(\mathcal{E}, \mathcal{B}||\mathcal{A}_2)(C_2) \).

We can restate classic theorem of composability of implementation in a quite general form.

**Theorem 4.20 (Implementation Composability).** Let \( f(\_\_\_\_\_\_) \) be an insight-function stable by composition. Let \( \text{Sch} \) be a scheduler schema. Let \( \mathcal{A}_1, \mathcal{A}_2, \mathcal{B} \) be PSIOA, s. t. \( \mathcal{A}_1 \leq_{\text{Sch}}^f \mathcal{A}_2 \). If \( \mathcal{B} \) is partially compatible with \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) then \( \mathcal{B}||\mathcal{A}_1 \leq_{\text{Sch}}^f \mathcal{B}||\mathcal{A}_2 \).

**Proof.** If \( \mathcal{E} \) is an environment for both \( \mathcal{B}||\mathcal{A}_1 \) and \( \mathcal{B}||\mathcal{A}_2 \), then \( \mathcal{E}' = \mathcal{E}||\mathcal{B} \) is an environment for both \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \). By associativity of parallel composition, we have for every \( i \in \{1, 2\} \), \( (\mathcal{E}||\mathcal{B})||\mathcal{A}_i = \mathcal{E}||((\mathcal{B}||\mathcal{A}_i)) \). Since \( \mathcal{A}_1 \leq_{\text{Sch}}^f \mathcal{A}_2 \), we have that \( \mathcal{B}||\mathcal{A}_1 \leq_{\text{Sch}}^f \mathcal{B}||\mathcal{A}_2 \). Therefore, \( \mathcal{B}||\mathcal{A}_1 \leq_{\text{Sch}}^f \mathcal{B}||\mathcal{A}_2 \).
\( \mathcal{A}_2 \), for any scheduler \( \sigma \in \text{Sch}(\mathcal{E}\|\mathcal{B})\|\mathcal{A}_1) \), it exists a corresponding scheduler \( \sigma' \in \text{Sch}(\mathcal{E}\|\mathcal{B})\|\mathcal{A}_2) \), s. t. \( f\text{-dist}(\mathcal{E}\|\mathcal{B},\mathcal{A}_1)(\epsilon_\sigma) = f\text{-dist}(\mathcal{E}\|\mathcal{B},\mathcal{A}_2)(\epsilon_\sigma') \). Thus, by stability by composition, for any scheduler \( \sigma \in \text{Sch}(\mathcal{E}\|\mathcal{B})\|\mathcal{A}_1) \), it exists a corresponding schedule \( \sigma' \in \text{Sch}(\mathcal{E}\|\mathcal{B})(\mathcal{A}_2) \), s. t. \( f\text{-dist}(\mathcal{E},\mathcal{B}\|\mathcal{A}_1)(\epsilon_\sigma) = f\text{-dist}(\mathcal{E},\mathcal{B}\|\mathcal{A}_2)(\epsilon_\sigma') \), that is \( \mathcal{A}_1\|\mathcal{B} \leq_{\text{Sch}} \mathcal{A}_2\|\mathcal{B} \).

Now we introduce the insight function \( \text{print}_i(\mathcal{E},\mathcal{A}) \) that we will use for monotonicity of implementation w.r.t. PSIOA creation.

**Definition 4.21 (print\(_i(\mathcal{E},\mathcal{A})\)).** Let \( \mathcal{A} \) be a PSIOA and \( \mathcal{E} \in \text{env}(\mathcal{A}) \) We note

\[
\text{print}_i(\mathcal{E},\mathcal{A}) : \begin{cases} 
\text{Frags}(\mathcal{E}||\mathcal{A}) & \to & \text{Frags}(\mathcal{E}) \times \text{trace}(\mathcal{A}) \\
\alpha & \mapsto & (\alpha \uparrow \mathcal{E}, \text{trace}(\alpha))
\end{cases}
\]

We note \( \text{Prints}(\mathcal{E},\mathcal{A}) \triangleq \text{range}(\text{print}_i(\mathcal{E},\mathcal{A})) \) and \( \forall (e, \beta, (e', \beta')) \in \text{Prints}(\mathcal{E},\mathcal{A})^2 \), \( (e, \beta) \leq (e', \beta') \) iff both \( e \leq e' \) and \( \beta \leq \beta' \). For every \( \zeta \in \text{Prints}(\mathcal{E},\mathcal{A}) \), we note \( \mathcal{C}_\zeta = \{ \zeta' \in \text{Prints}(\mathcal{E},\mathcal{A}) \mid \zeta \leq \zeta' \} \) called a cone of prints. We note \( \mathcal{F}_{\text{Prints}}(\mathcal{E},\mathcal{A}) \) the \( \sigma \)-field generated by the set of cones of prints.

**Lemma 4.22 (Print is an Insight Function Stable by Composition).** Let \( \mathcal{A} \) be a PSIOA and \( \mathcal{E} \in \text{env}(\mathcal{A}) \), \( \text{print}_i(\mathcal{E},\mathcal{A}) \) is an insight function stable by composition, a measurable function from \( \mathcal{F}_{\text{Execs}}(\mathcal{E}|\mathcal{A}) \) to \( \mathcal{F}_{\text{Frags}}(\mathcal{E}|\mathcal{A}) \).

**Proof.**

1. (measurability) We need to show that \( \forall G \in \mathcal{F}_{\text{Prints}}(\mathcal{E},\mathcal{A}), \text{print}_i^{-1}(\mathcal{E},\mathcal{A})_i(G) \in \mathcal{F}_{\text{Frags}}(\mathcal{E}|\mathcal{A}) \).

   We note \( f_1 : \alpha \in \text{Frags}(\mathcal{E}||\mathcal{A}) \mapsto \alpha \uparrow \mathcal{E} \) and \( f_2 : \alpha \in \text{Frags}(\mathcal{E}||\mathcal{A}) \mapsto \text{trace}(\alpha) \). We can already remark that 
   
   (*) \( \forall i \in \{1,2\}, \forall (a, a') \in \text{Frags}^2(\mathcal{A}), a \leq a' \Rightarrow f_i(a) \leq f_i(a') \) and 
   
   (**) \( \forall i \in \{1,2\}, \forall (y_i, a') \in \text{range}(f_i) \times \text{Frags}(\mathcal{A}), y_i \leq f_i(a') \Rightarrow \exists (a, a'') \in \text{Frags}^2(\mathcal{A}), a' = a'' \) and \( f_i(a) = y_i \).

   Let \( G \in \mathcal{F}_{\text{Prints}}(\mathcal{E},\mathcal{A}) \). By construction, it exists \( \zeta = (e, \beta) \in \text{Prints}(\mathcal{E},\mathcal{A}) \) s. t. \( G = \mathcal{C}_\zeta \). Let \( F \triangleq \text{print}_i^{-1}(\mathcal{E},\mathcal{A})_i(G) = \{ a' \in \text{Frags}(\mathcal{A}) \mid \zeta \leq \text{print}_i(\mathcal{E},\mathcal{A})(a') \} = F_1 \cap F_2 \) with \( F_1 = \{ a'_1 \in \text{Frags}(\mathcal{A}) \mid e \leq f_i(a'_1) \} \) and \( F_2 = \{ a'_2 \in \text{Frags}(\mathcal{A}) \mid \beta \leq f_i(a'_2) \} \). By (*) and (**) \( F_1 = \bigcup_{a_1 \in \text{Frags}(\mathcal{E})} a_1 \downarrow \mathcal{E} = c_{a_1} \) and \( F_2 = \bigcup_{a_2 \in \text{Frags}(\mathcal{A})} \text{trace}(a_2) = \beta \). By closeness of \( \sigma \)-field under countable union, \( F_1, F_2 \in \mathcal{F}_{\text{Frags}}(\mathcal{E}|\mathcal{A}) \) and by closeness of \( \sigma \)-field under intersection \( F \in \mathcal{F}_{\text{Frags}}(\mathcal{E}|\mathcal{A}) \), which ends the proof.

2. (stability by composition) Let \( (\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}, \mathcal{E}) \) be a quadruplet of PSIOA, s. t. \( \mathcal{B} \) is compatible with \( \mathcal{A}_1 \) and \( \mathcal{A}_2, \mathcal{E} \in \text{env}(\mathcal{B}||\mathcal{A}_1) \cap \text{env}(\mathcal{B}||\mathcal{A}_2) \). Let \( (a, \pi) \in \text{Execs}_{\mathcal{E}||\mathcal{B}}(\mathcal{A}_1) \times \text{Execs}_{\mathcal{E}||\mathcal{B}}(\mathcal{A}_2) \), clearly \( \alpha \uparrow (\mathcal{E}||\mathcal{B}) \mapsto \alpha \uparrow (\mathcal{E}||\mathcal{B}) \) and \( \text{trace}(\alpha) \) stay the same. Thus, for every \( (C_1, C_2) \in \mathcal{F}_{\text{Execs}}(\mathcal{E}||\mathcal{B})||\mathcal{A}_1) \times \mathcal{F}_{\text{Execs}}(\mathcal{E}||\mathcal{B})||\mathcal{A}_2) \), \( \text{print}_i(\mathcal{E}||\mathcal{B},\mathcal{A}_1)(C_1) = \text{print}_i(\mathcal{E}||\mathcal{B},\mathcal{A}_2)(C_2) \Rightarrow f_i(\mathcal{E}||\mathcal{B})(C_1) = f_i(\mathcal{E}||\mathcal{B})(C_2) \).

Thus, given an environment \( \mathcal{E} \) of \( \mathcal{A} \) probability measure \( \mu \) on \( \mathcal{F}_{\text{Execs}}(\mathcal{E}||\mathcal{A}) \), and a scheduler \( \sigma \) of \( (\mathcal{E}||\mathcal{A}) \) we define \( \text{p dist}_i(\mathcal{E},\mathcal{A})(\sigma, \mu, \mu) \), to be the image measure of \( \epsilon_{\sigma,\mu} \) under \( \text{print}_i(\mathcal{E},\mathcal{A}) \). We note \( \text{p dist}_i(\mathcal{E},\mathcal{A})(\sigma) \) for \( \text{p dist}_i(\mathcal{E},\mathcal{A})(\sigma, \delta_{\text{start}}(\mathcal{E},\mathcal{A})) \).

This choice that slightly differs from \( t\text{dist}(\mathcal{E},\mathcal{A})(\sigma, \mu, \mu) = \text{trace-dist}(\mathcal{E},\mathcal{A})(\sigma, \mu) \) used in [4], is motivated by the achievement of monotonicity of \( \text{print} - \text{implementation} \) w.r.t. PSIOA creation.

### 4.7 Hiding operator

We anticipate the definition of configuration automata by introducing the classic hiding operator. This operator "hide" the output actions transforming them into internal actions.
Definition 4.23 (hiding on signature). Let $\text{sig} = (in, out, int)$ be a signature and $acts$ a set of actions. We note $\text{hide}(\text{sig}, acts)$ the signature $\text{sig'} = (in', out', int')$ s.t.

- $in' = in$
- $out' = out \setminus acts$
- $int' = int \cup (out \cap acts)$

Definition 4.24 (hiding on PSIOA). Let $\mathcal{A} = (\mathcal{Q}, \mathcal{R}, \bar{q}, \mathcal{A}, D, \mathcal{R})$ be a PSIOA. Let $h$ a function mapping each state $q \in Q$ to a set of output actions. We note $\text{hide}(\mathcal{A}, h)$ the PSIOA $(\bar{Q}, \bar{q}, \mathcal{A}, D)$, where $\text{sig}'(\mathcal{A}) : q \in Q \mapsto \text{hide}(\text{sig}(\mathcal{A})(q), h(q))$.

Lemma 4.25 (Hiding and composition are commutative). Let $\text{sig}_a = (in_a, out_a, int_a)$, $\text{sig}_b = (in_b, out_b, int_b)$ be compatible signature and $acts_a\!,\! acts_b$ some set of actions, s.t.

- $(acts_a \cap out_a) \cap \bar{\text{sig}}_b = \emptyset$ and
- $(acts_b \cap out_b) \cap \bar{\text{sig}}_a = \emptyset$.

then $\text{sig}'_a \! \times \! \text{sig}'_b \! = \! \text{hide}(\text{sig}_a \times \text{sig}_b, acts_a \cup acts_b)$. 

Proof.

- Compatibility: After hiding operation, we have:
  - $\text{in}'_a = in_a, \text{in}'_b = in_b$
  - $\text{out}'_a = out_a \setminus acts_a, \text{out}'_b = out_b \setminus acts_b$
  - $\text{int}'_a = int_a \cup (out_a \cap acts_a), \text{int}'_b = int_b \cup (out_b \cap acts_b)$

Since $\text{out}_a \cap \text{out}_b = \emptyset$, a fortiori $\text{out}'_a \cap \text{out}'_b = \emptyset, \text{int}_a \cap \bar{\text{sig}}_b = \emptyset$, thus if $(\text{out}_a \cap acts_a) \cap \bar{\text{sig}}_b = \emptyset$, then $\text{int}'_a \cap \bar{\text{sig}}_b = \emptyset$ and with the symmetric argument, $\text{int}'_b \cap \bar{\text{sig}}_a = \emptyset$. Hence, $\text{sig}'_a$ and $\text{sig}'_b$ are compatible.

- Commutativity:

  After composition of $\text{sig}'_c = \text{sig}'_a \times \text{sig}'_b$ operation, we have:
  - $\text{out}'_c = \text{out}'_a \cup \text{out}'_b = (\text{out}_a \setminus acts_a) \cup (\text{out}_b \setminus acts_b)$. If $\text{out}_b \cap acts_a = \emptyset$ and $\text{out}_a \cap acts_b = \emptyset$, then
    - $\text{out}'_c = (\text{out}_a \cup \text{out}_b) \\setminus (acts_a \cup acts_b)$.
  - $\text{in}'_c = \text{in}'_a \cup \text{in}'_b \setminus \text{out}'_c$ = $in_a \cup in_b \setminus \text{out}'_c$
  - $\text{int}'_c = \text{int}'_a \cup \text{int}'_b = int_a \cup (\text{out}_a \cap acts_a) \cup (\text{out}_b \cap acts_b) = int_a \cup int_b \cup (\text{out}_a \cap acts_a) \cup (\text{out}_b \cap acts_b)$.

If $\text{out}_b \cap acts_a = \emptyset$ and $\text{out}_a \cap acts_b = \emptyset$, then $\text{int}'_c = int_a \cup int_b \cup (\text{out}_a \cup \text{out}_b) \cap (acts_a \cup acts_b)$.

and after composition of $\text{sig}_d = \text{sig}_a \times \text{sig}_b$

- $\text{out}_d = \text{out}_a \cup \text{out}_b$
- $\text{in}_d = \text{in}_a \cup \text{in}_b \setminus \text{out}_d$
- $\text{int}_d = \text{int}_a \cup \text{int}_b$

Finally, after hiding operation $\text{sig}'_d = \text{hide}(\text{sig}_d, acts_a \cup acts_b)$ we have:

- $\text{in}'_d = \text{in}_d$
- $\text{out}'_d = \text{out}_d \setminus acts_a \cup acts_b = (\text{out}_a \cup \text{out}_b) \setminus (acts_a \cup acts_b)$
- $\text{int}'_d = \text{int}_d \cup (\text{out}_d \cap (acts_a \cup acts_b)) = (\text{int}_a \cup \text{int}_b) \cup (\text{out}_d \cap (acts_a \cup acts_b))$
We combine the notion of configuration of [1] with the probabilistic setting of [8]. A configuration is a set of automata.

We anticipate the definition of isomorphism between PSIOA that differs only syntactically.

**Remark 1.** We can restrict hiding operation to set of actions included in the set of output actions of the signature (act ⊆ out). In this case, since we already have out_a ∩ act_b = Ø by compatibility, we immediately have out_a ∩ act_b = Ø and out_b ∩ act_a = Ø. Thus to obtain compatibility, we only need in_a ∩ act_b = Ø and in_b ∩ act_a = Ø. Later, the compatibility of PCA will implicitly assume this predicate (otherwise the PCA could not be compatible).

### 4.8 State renaming operator

We anticipate the definition of isomorphism between PSIOA that differs only syntactically.

**Definition 4.26.** (State renaming for PSIOA) Let A be a PSIOA with Q,A as set of states, let Q,A’ be another set of states and let ren : Q,A → Q,A’ be a bijective mapping. Then ren(A) (we abuse the notation) is the automaton given by:

- \( \text{start}(\text{ren}(A)) = \text{ren}(\text{start}(A)) \)
- \( \text{states}(\text{ren}(A)) = \text{ren}(\text{states}(A)) \)
- \( \forall q_A \in \text{states}(\text{ren}(A)), \text{sig}(\text{ren}(A))(q_A) = \text{sig}(A)(\text{ren}^{-1}(q_A)) \)
- \( \forall q_A' \in \text{states}(\text{ren}(A)), \forall a \in \text{sig}(\text{ren}(A))(q_A'), \text{if } (\text{ren}^{-1}(q_A'), a, \eta) \in D_A, \text{then } (q_A', a, \eta') \in D_{\text{ren}(A)} \)

where \( \eta' \in \text{Disc}(Q,A', F_{Q,A'}) \) and for every \( q_{A'}' \in \text{states}(\text{ren}(A)), \eta'(q_{A'}') = \eta(\text{ren}^{-1}(q_{A'}')). \)

**Definition 4.27.** (State renaming for PSIOA execution) Let A and A’ be two PSIOA s.t. A’ = ren(A). Let \( \alpha = q_0 a_1 q_1 \ldots \) be an execution fragment of A. We note ren(\( \alpha \)) the sequence \( \text{ren}(q_0) a_1 \text{ren}(q_1) \ldots \)

**Lemma 4.28.** Let A and A’ be two PSIOA s.t. A’ = ren(A) with ren : states(A) → states(A’) being a bijective map. Let \( \alpha \) be an execution fragment of A. The sequence ren(\( \alpha \)) is an execution fragment of A.

**Proof.** Let \( q_1 a_1 q_1 a_1 q_1 \ldots \) be a subsequence of \( \alpha \). ren(\( q_1 \)) ∈ states(A) by definition, \( a_1 \in \text{sig}(A')(\text{ren}(q_1)) \) since \( \text{sig}(A')(\text{ren}(q_1)) = \text{sig}(A)(q_1) \), \( \eta(A', \text{ren}(q_1), a_1) \) \( \text{ren}(q_1) a_1 q_1 \ldots \) is \( \eta(A, q_1 a_1 q_1 \ldots) > 0. \)

### 5 PROBABILISTIC CONFIGURATION AUTOMATA

We combine the notion of configuration of [1] with the probabilistic setting of [8]. A configuration is a set of automata attached with their current states. This will be a very useful tool to define dynamicty by mapping the state of an automaton of a certain “layer” to a configuration of automata of lower layer, where the set of automata in the configuration can dynamically change from on state of the automaton of the upper level to another one.

#### 5.1 configuration

**Definition 5.1 (Configuration).** A configuration is a pair (A, S) where

- A = \{A_1, ..., A_n\} is a finite set of PSIOA identifiers and
- S maps each A_k ∈ A to an s_k ∈ states(A_k).
In distributed computing, configuration usually refers to the union of states of all the automata of the "system". Here, there is a subtlety, since it captures a set of some automata \( A \) in their current state \( S \), but the set of automata of the systems will not be fixed in the time.

**Definition 5.2 (Compatible configuration).** A configuration \( (A,S) \), with \( A = \{ \mathcal{A}_1, ..., \mathcal{A}_n \} \), is compatible iff the set \( A \) is compatible at state \( (S(\mathcal{A}_1), ..., S(\mathcal{A}_n)) \) as per definition 4.6.

**Definition 5.3 (Intrinsic attributes of a configuration).** Let \( C = (A,S) \) be a compatible configuration. Then we define

- \( \text{aus}(C) = A \) represents the automata of the configuration,
- \( \text{map}(C) = S \) maps each automaton of the configuration with its current state,
- \( \text{out}(C) = \bigcup_{\mathcal{A} \in A} \text{out}(\mathcal{A})(S(\mathcal{A})) \) represents the output actions of the configuration,
- \( \text{in}(C) = \left( \bigcup_{\mathcal{A} \in A} \text{in}(\mathcal{A})(S(\mathcal{A})) \right) - \text{out}(C) \) represents the input actions of the configuration,
- \( \text{int}(C) = \bigcup_{\mathcal{A} \in A} \text{int}(\mathcal{A})(S(\mathcal{A})) \) represents the internal actions of the configuration,
- \( \text{ext}(C) = \text{in}(C) \cup \text{out}(C) \) represents the external actions of the configuration,
- \( \text{sig}(C) = (\text{in}(C), \text{out}(C), \text{int}(C)) \) is called the intrinsic signature of the configuration,
- \( US(C) = (S(\mathcal{A}_1), ..., S(\mathcal{A}_n)) \) represents the states of the set of automata of the configuration.

Here we define a reduced configuration as a configuration deprived of the automata that are in the very particular state where their current signatures are the empty set. This mechanism will be used later to capture the idea of destruction of an automaton.

**Definition 5.4 (Reduced configuration).** \( \text{reduce}(C) = (A', S') \), where \( A' = \{ \mathcal{A} | \mathcal{A} \in A \text{ and } \text{sig}(\mathcal{A})(S(\mathcal{A})) \neq \emptyset \} \) and \( S' \) is the restriction of \( S \) to \( A' \), noted \( S \restriction A' \) in the remaining.

A configuration \( C \) is a reduced configuration iff \( C = \text{reduce}(C) \).

### 5.2 Configuration transition

We will define some probabilistic transition from configurations to others where some automata can be destroyed or created. To do it properly, we start by defining "preserving transition" where no automaton is neither created nor destroyed and then we define above this definition the notion of configuration transition. These distributions belong to the measurable set \( \{ Q_{conf}.\text{Disc}(Q_{conf}) \} \) where \( Q_{conf} \) denotes the (countable) set of configurations.

**Lemma 5.5.** The set \( Q_{conf} \) of configurations is countable.

**Proof.** (1) \( \{ A \in \mathcal{P}(\text{Autids}) | A \text{ is finite } \} \) is countable, (2) \( \forall \mathcal{A} \in \text{Autids} \), \( \text{states}(\mathcal{A}) \) is countable by definition 4.1 of PSIOA and (3) the cartesian product of countable sets is a countable set.

**Definition 5.6 (Preserving distribution).** A preserving distribution \( \eta_p \in \text{Disc}(Q_{conf}) \) s.t. it exists a set of automata \( A \), called automata support of \( \eta_p \), such that \( \forall (A',S') \in \text{supp}(\eta_p), A = A' \).

We define a companion distribution as the natural distribution of the corresponding set of automata at the corresponding current state. Since no creation or destruction occurs, these definitions can seem redundant, but this is only an intermediate step to define properly the "dynamic" distribution.

**Definition 5.7 (Companion distribution).** Let \( C = (A,S) \) be a compatible configuration with \( A = \{ \mathcal{A}_1, ..., \mathcal{A}_n \} \) and \( S : \mathcal{A}_i \in A \mapsto q_i \in Q_{\mathcal{A}_i} \) (with \( A \) partially-compatible at state \( q = (q_1, ..., q_n) \in Q_A = Q_{\mathcal{A}_1} \times \cdots \times Q_{\mathcal{A}_n} \)). Let \( \eta_p \) be a
preserving distribution with \(A\) as automata support. The probabilistic distribution \(\eta_{(A,q,a)}\) is a companion distribution of \(\eta_p\) if for every \(q' = (q'_1, \ldots, q'_n) \in Q_A\), for every \(S'' : A_i \in A \mapsto q''_i \in Q_i\),

if for every configuration \(C''\), for every \(q' \in Q_A\), \(US(C'') = q' \implies \eta_{(A,q,a)}(q') = \eta_p(C'')\).

That is the distribution \(\eta_{(A,q,a)}\) corresponds exactly to the distribution \(\eta_p\).

This is "a" and not "the" companion distribution since \(\eta_p\) does not explicit the start configuration. So \(\eta_p\) can have several companion distributions.

**Fig. 14.** A preserving distribution is matching its companion distribution.

Now, we can naturally define a preserving transition \((C,a,\eta_p)\) from a configuration \(C\) via an action \(a\) with a companion transition of \(\eta_p\). It allows us to say what is the "static" probabilistic transition from a configuration \(C\) via an action \(a\) if no creation or destruction occurs.

**Definition 5.8 (preserving transition).** Let \(C = (A,S)\) be a compatible configuration, \(q = US(C)\) and \(\eta_p \in Disc(Q_{conf})\) be a preserving transition with \(A_s\) as automata support.

Then we say that \((C,a,\eta_p)\) is a preserving configuration transition, noted \(C \xrightarrow{\alpha} \eta_p\) if

- \(A_s = A\)
- \(\eta_{(A,q,a)}\) is a companion distribution of \(\eta_p\)

For every preserving configuration transition \((C,a,\eta_p)\), we note \(\eta_{((C,a),p)} = \eta_p\).

The preserving transition of a configuration corresponds to the transition of the composition of the corresponding automata at their corresponding current states.

Now we are ready to define our "dynamic" transition, that allows a configuration to create or destroy some automata.

At first, we define reduced distribution that leads to reduced configurations only, where all the automata that reach a state with an empty signature are destroyed.

**Definition 5.9 (reduced distribution).** A reduced distribution \(\eta_r \in Disc(Q_{conf})\) is a probabilistic distribution verifying that for every configuration \(C \in supp(\eta_r)\), \(C = reduced(C)\).

Now, we generate reduced distribution with a preserving distribution that describes what happen to the automata that already exist and a set of new automata that are created.
Definition 5.10 (Generation of reduced distribution). Let \( \eta_p \in \text{Disc}(Q_{\text{conf}}) \) be a preserving distribution with \( A \) as automata support. Let \( \varphi \subseteq \text{Autids} \), \( \varphi \) is finite. We say the reduced distribution \( \eta_r \in \text{Disc}(Q_{\text{conf}}) \) is generated by \( \eta_p \) and \( \varphi \) if it exists a non-reduced distribution \( \eta_{nr} \in \text{Disc}(Q_{\text{conf}}) \), s.t.

- (\( \varphi \) is created with probability 1)
  \[ \forall (A'', S'') \in \text{supp}(\eta_{nr}), A'' = A \cup \varphi. \]
- (freshly created automata start at start state)
  \[ \forall (A', S') \in \text{supp}(\eta_{nr}), \forall A_i \in \varphi \setminus A, S''(A_i) = \text{start}(A_i). \]
- (The non-reduced transition match the preserving transition)
  \[ \forall (A'', S'') \in Q_{\text{conf}}, \text{s.t.} \ A'' = A \cup \varphi \text{ and } \forall A_i \in \varphi \setminus A, S''(A_i) = \text{start}(A_i), \eta_{nr}((A'', S'')) = \eta_p(A, S''[A]) \]
  where \( S''[A] \) denotes the restriction of \( S'' \) on \( A \)
- (The reduced transition match the non-reduced transition)
  \[ \forall c' \in Q_{\text{conf}}, \text{if } c' = \text{reduce}(c'), \eta_r(c') = \Sigma_{c'' = \text{reduce}(c')} \eta_{nr}(c''), \text{if } c' \neq \text{reduce}(c'), \text{then } \eta_r(c') = 0 \]

Definition 5.11 (Intrinsic transition). Let \( (A, S) \) be arbitrary reduced compatible configuration, let \( \eta \in \text{Disc}(Q_{\text{conf}}) \), and let \( \varphi \subseteq \text{Autids} \), \( \varphi \cap A = \emptyset \) and \( \varphi \) is finite. Then \( (A, S) \xrightarrow{\alpha} \varphi \eta \) if \( \eta \) is generated by \( \eta_p \) and \( \varphi \) with \( (A, S) \xrightarrow{\alpha} \eta_p \).

\[
C = (A, S) \quad \varphi = \{A_1\} \quad A' = (A_2, A_3, A_4)
\]

\[
C_1 = (A', S_1') \quad A = (A_1, A_2, A_3) \quad \eta \quad \alpha \\
\downarrow \quad \eta \\
C_2 = (A', S_2')
\]

\[ C \xrightarrow{n} \varphi \eta \]

Fig. 15. An intrinsic transition where \( A_1 \) is destroyed deterministically and \( A_4 \) is created deterministically.

The assumption of deterministic creation is not restrictive, nothing prevents from flipping a coin at state \( s_0 \) to reach \( s_1 \) with probability \( p \) or \( s_2 \) with probability \( 1 - p \) and only create a new automaton in state \( s_2 \) with probability 1, while the action create is not enabled in state \( s_1 \).

5.3 Probabilistic Configuration Automata

Here we define our probabilistic configuration automata. Just before that, we introduce a notation to represents corresponding probability measures whose respective supports are linked by a bijection that preserves the measure.

Definition 5.12 (\( \eta \xrightarrow{f} \eta' \)). Let \( Q \) and \( Q' \) be two countable sets. Let \( (\eta, \eta') \in \text{Disc}(Q) \times \text{Disc}(Q') \). Let \( f : Q \rightarrow Q' \). We note \( \eta \xrightarrow{f} \eta' \) if the following is verified:

\[
(\eta \xrightarrow{f} \eta') \Leftrightarrow \forall (A, S) \in \text{supp}(\eta), \exists (A', S') \in \text{supp}(\eta'), A' = \text{reduce}(A), S'(A') = \text{start}(A)
\]
• the restriction \( \tilde{f} \) of \( f \) to \( \text{supp}(\eta) \) is a bijection from \( \text{supp}(\eta) \) to \( \text{supp}(\eta') \)

\[ \forall q \in \text{supp}(\eta), \eta(q) = \eta'(f(q)) \]

**Lemma 5.13.** \( \eta \xrightarrow{f} \eta' \) and \( \eta' \xleftarrow{g} \eta'' \) implies

\[ \eta' \xrightarrow{k} \eta \] where the restriction \( \tilde{h} \) of \( h \) on \( \text{supp}(\eta) \) verifies \( \tilde{h} = g \circ \tilde{f} \) and

\[ \eta'' \xleftarrow{k'} \eta' \] where the restriction \( \tilde{k} \) to \( \text{supp}(\eta') \) verifies \( \tilde{k} = \tilde{f}^{-1} \)

**Proof.** For the first item: The composition of two bijection is a bijection and the reverse function of a bijection is a bijection. For the second item: In the first case, \( \forall q \in \text{supp}(\eta), \eta(q) = \eta'(f(q)) \) with \( f(q) \in \text{supp}(\eta') \) which means \( \eta'(f(q)) = \eta''(g(f(q))) \). In the second case \( \forall q' \in \text{supp}(\eta') \), \( \exists q \in \text{supp}(\eta), \eta(q) = \eta'(q') = \tilde{f}(q) \) and hence \( \forall q' \in \text{supp}(\eta'), \eta'(q') = \eta(q = \tilde{f}^{-1}(q')) \)

Now we are ready to define our probabilistic configuration automata. Such an automaton define a strong link with a dynamic configuration.

**Definition 5.14 (Probabilistic Configuration Automaton).** A probabilistic configuration automaton (PCA) \( K \) consists of the following components:

1. A probabilistic signature I/O automaton \( \text{psioa}(K) \). For brevity, we define \( \text{states}(K) = \text{states}(\text{psioa}(K)), \text{start}(K) = \text{start}(\text{psioa}(K)), \text{sig}(K) = \text{sig}(\text{psioa}(K)), \text{steps}(K) = \text{steps}(\text{psioa}(K)) \), and likewise for all other (sub)components and attributes of \( \text{psioa}(K) \).

2. A configuration mapping \( \text{config}(K) \) with domain \( \text{states}(K) \) and such that \( \text{config}(K)(q_K) \) is a reduced compatible configuration for all \( q_K \in \text{states}(K) \).

3. For each \( q_K \in \text{states}(K) \), a mapping \( \text{created}(K)(q_K) \) with domain \( \text{sig}(K)(q_K) \) and such that \( \forall a \in \text{sig}(K)(q_K) \), \( \text{created}(K)(q_K)(a) \subseteq \text{Autids} \) with \( \text{created}(K)(q_K)(a) \) finite.

4. A hidden-actions mapping \( \text{hidden-actions}(K) \) with domain \( \text{states}(K) \) and such that \( \text{hidden-actions}(K)(q_K) \subseteq \text{out}(\text{config}(K)(q_K)) \).

and satisfies the following constraints:

1. (start states preservation) If \( \text{config}(K)(q_{K_a}) = (A, S) \), then \( \forall \mathcal{A}_i \in A, S(\mathcal{A}_i) = \hat{q}_i \)

2. (top/down transition preservation) If \( (q_K, a, \eta_{(K_a,q_K,a)}) \in \text{dtrans}(K) \) then it exists \( \eta' \in \text{Disc}(Q_{\text{conf}}) \) s. t. \( \eta_{(K_a,q_K,a)} \xrightarrow{f} \eta' \) with i) \( f = \text{config}(K) \) and ii) \( \text{config}(K)(q_K) \xrightarrow{a} \eta' \), where \( \phi = \text{created}(K)(q_K)(a) \)

3. (bottom/up transition preservation) If \( q_K \in \text{states}(K) \) and \( \text{config}(K)(q_K) \xrightarrow{a} \eta' \) for some action \( a, \phi = \text{created}(K)(x)(a) \), and reduced compatible probabilistic measure \( \eta' \in \text{Disc}(Q_{\text{conf}}) \), then \( (q_K, a, \eta_{(K_a,q_K,a)}) \in \text{dtrans}(K) \), and \( \eta_{(K_a,q_K,a)} \xrightarrow{f} \eta' \) with \( f = \text{config}(K) \).

4. (signature preservation modulo hiding) For all \( q_K \in \text{states}(K), \text{sig}(K)(q_K) = \text{hide}(\text{config}(K)(q_K)), \text{hidden-actions}(q_K) \), which implies that

- (a) \( \text{out}(K)(q_K) \subseteq \text{out}(\text{config}(K)(q_K)) \),
- (b) \( \text{in}(K)(q_K) = \text{in}(\text{config}(K)(q_K)) \),
- (c) \( \text{int}(K)(q_K) \supseteq \text{int}(\text{config}(K)(q_K)) \), and
- (d) \( \text{out}(K)(q_K) \cup \text{int}(X)(q_K) = \text{out}(\text{config}(K)(q_K)) \cup \text{int}(\text{config}(K)(q_K)) \)

This definition, proposed in a deterministic fashion in [1], captures dynamicity of the system. Each state is linked with a configuration. The set of automata of the configuration can change during an execution. A sub-automaton \( \mathcal{A} \)
is created from state $q$ by the action $a$ if $A \in \text{created}(K)(q)(a)$. A sub-automaton $A$ is destroyed if the non-reduced attached configuration distribution lead to a configuration where $A$ is in a state $q^A_s$, i.e., $\text{sign}(A)(q^A_s) = \emptyset$. Then the corresponding reduced configuration will not hold $A$. The last constraint states that the signature of a state $q_K$ of $K$ must be the same as the signature of its corresponding configuration $\text{config}(K)(q_K)$, except for the possible effects of hiding operators, so that some outputs of $\text{config}(K)(q_K)$ may be internal actions of $K$ in state $q_K.$

**Definition 5.15 (hiding on PCA).** Let $X$ be a PCA. Let $h : q \in \text{states}(X) \mapsto h(q) \subseteq \text{out}(X)(q)$ a function mapping each state $q \in \text{states}(X)$ to a set of output actions. We note $\text{hide}(X, h)$ the PCA $X'$ that differs from $X$ only on $\text{sig}(X')$ and $\text{hidden-actions}(X')$, where $\forall q \in \text{states}(X) = \text{states}(X'),$

- $\text{sig}(X')(q) = \text{hide}(\text{sig}(X)(q), h(q))$ and
- $\text{hidden-actions}(X')(q) = \text{hidden-actions}(X)(q) \cup h(q)$.

Additionally, we recursively define the current constitution of a PCA. To do so, we assume the existence of a subset $\text{Autids}_0 \subseteq \text{Autids}$ that represents the "atomic entities" of our formalism. Any automaton is constructed with automata in $\text{Autids}_0$. 

---

**Fig. 16.** A PCA life cycle.
Definition 5.16 (Constitution). For every $\mathcal{A} \in \text{Autids}$, we note $\text{constitution}(\mathcal{A}) : \text{states}(\mathcal{A}) \rightarrow \mathcal{P}(\text{Autids}_0)$ s.t.

- $\forall \mathcal{A} \in \text{Autids}_0, \forall q \in \text{states}(\mathcal{A})$, $\text{constitution}(\mathcal{A})(q) = \{ q \}$.
- $\forall \mathcal{A} = (\mathcal{A}_1, ..., \mathcal{A}_n) \in (\text{Autids}_0)^n$, $\forall q \in \text{states}(\mathcal{A})$ with $\mathcal{A} = \mathcal{A}_1||...||\mathcal{A}_n$, $\text{constitution}(\mathcal{A})(q) = \mathcal{A}$.

the constitution of a PCA is defined recursively through its configuration, i.e. for every PCA $X$, $\forall q \in \text{states}(X)$, if we note $(A,S) = \text{config}(X)(q)$, $\text{constitution}(X)(q) = \bigcup_{\mathcal{A} \in A} \text{constitution}(\mathcal{A})(S(\mathcal{A}))$.

We note $UA(K) = \bigcup_{q \in \text{states}(K)} (\text{constitution}(K)(q))$.

5.4 Compatibility, composition

Again, we want to define composition operator for PCA to captures the fact the comportment of one PCA can influence the comportment of another PCA. Some syntaxic rules have to be defined first.

Compatibility and union of configuration. We extend the formalism of configuration with the union operator. We do not need a composition operator to prevent from think such an union is always compatible, which is not true.

Definition 5.17 (Union of configurations). Let $C_1 = (A_1,S_1)$ and $C_2 = (A_2,S_2)$ be configurations such that $A_1 \cap A_2 = \emptyset$. Then, the union of $C_1$ and $C_2$, denoted $C_1 \cup C_2$, is the configuration $C = (A,S)$ where $A = A_1 \cup A_2$ and $S$ agrees with $S_1$ on $A_1$, and with $S_2$ on $A_2$. It is clear that configuration union is commutative and associative. Hence, we will freely use the $n$-ary notation $C_1 \cup ... \cup C_n$ (for any $n \geq 1$) whenever $\forall i, j \in [1 : n], i \neq j, \text{auts}(C_i) \cap \text{auts}(C_j) = \emptyset$.

Lemma 5.18. Let $C_1 = (A_1,S_1)$ and $C_2 = (A_2,S_2)$ be compatible configurations such that $A_1 \cap A_2 = \emptyset$. Let $C = (A,S) = C_1 \cup C_2$ be a compatible configuration. Then $\text{sig}(C) = \text{sig}(C_1) \times \text{sig}(C_2)$

Proof.

- $\text{out}(C) = \bigcup_{\mathcal{A}_k \in A} \text{out}(\mathcal{A}_k)(S(\mathcal{A}_k)) = (\bigcup_{\mathcal{A}_1 \in A_1} \text{out}(\mathcal{A}_1)(S(\mathcal{A}_1))) \cup (\bigcup_{\mathcal{A}_2 \in A_2} \text{out}(\mathcal{A}_2)(S(\mathcal{A}_2))) = \text{out}(C_1) \cup \text{out}(C_2)$
- $\text{in}(C) = \bigcup_{\mathcal{A}_k \in A} \text{in}(\mathcal{A}_k)(S(\mathcal{A}_k)) \setminus \text{out}(C) = (\bigcup_{\mathcal{A}_1 \in A_1} \text{in}(\mathcal{A}_1)(S(\mathcal{A}_1))) \cup (\bigcup_{\mathcal{A}_2 \in A_2} \text{in}(\mathcal{A}_2)(S(\mathcal{A}_2))) \setminus \text{out}(C) = \text{in}(C_1) \cup \text{in}(C_2)$
- $\text{int}(C) = \bigcup_{\mathcal{A}_k \in A} \text{int}(\mathcal{A}_k)(S(\mathcal{A}_k)) = (\bigcup_{\mathcal{A}_1 \in A_1} \text{int}(\mathcal{A}_1)(S(\mathcal{A}_1))) \cup (\bigcup_{\mathcal{A}_2 \in A_2} \text{int}(\mathcal{A}_2)(S(\mathcal{A}_2))) = \text{int}(C_1) \cup \text{int}(C_2)$

Thus $(\text{out}(C),\text{in}(C),\text{int}(C)) = (\text{out}(C_1) \cup \text{out}(C_2),\text{in}(C_1) \cup \text{in}(C_2))$

Definition 5.19 (PCA partially-compatible at a state). Let $X = \{X_1, ..., X_n\}$ be a set of PCA. We note $\text{psioa}(X) = \{\text{psioa}(X_1), ..., \text{psioa}(X_n)\}$. The PCA $X_1, ..., X_n$ are partially-compatible at state $q_X = (q_{X_1}, ..., q_{X_n}) \in \text{states}(X_1) \times ... \times \text{states}(X_n)$ iff:

1. Sub-automaton exclusivity: $\forall i,j \in [1 : n], i \neq j : \text{auts}(\text{config}(X_i)(q_{X_i})) \cap \text{auts}(\text{config}(X_j)(q_{X_j})) = \emptyset$.
2. Compatible signatures $\{\text{sig}(X_1)(q_{X_1}), ..., \text{sig}(X_n)(q_{X_n})\}$ is a set of compatible signatures.
3. Creation exclusivity: $\forall i,j \in [1 : n], i \neq j : \forall a \in \text{sig}(X_i)(q_{X_i}) \cup \text{sig}(X_j)(q_{X_j}) : \text{created}(X_i)(q_{X_i})(a) \cap \text{created}(X_j)(q_{X_j})(a) = \emptyset$.
4. Constitution exclusivity: $\forall i,j \in [1 : n], i \neq j : \text{constitution}(X_i)(q_{X_i}) \cap \text{constitution}(X_j)(q_{X_j}) = \emptyset$

We can remark that if $\forall i,j \in [1 : n], i \neq j : \text{auts}(\text{config}(X_i)(q_{X_i})) \cap \text{auts}(\text{config}(X_j)(q_{X_j})) = \emptyset$ and $\{\text{sig}(X_1)(q_{X_1}), ..., \text{sig}(X_n)(q_{X_n})\}$ is a set of compatible signatures, then $\text{config}(X_i)(q_{X_i}) \cup ... \cup \text{config}(X_n)(q_{X_n})$ is a reduced compatible configuration.
If $X$ is partially-compatible at state $q_X$, for every action $a \in \tilde{\sigma}(\text{psioa}(X))(q_X)$, we note $\eta(q_{Xq,a}) = \eta(\text{psioa}(X),q,a)$ and we extend this notation with $\eta(X,q_{X,a}) = \delta_{q_X}$ if $a \notin \tilde{\sigma}(\text{psioa}(X))(q_X)$.

**Definition 5.20 (pseudo execution).** Let $X = \{X_1, ..., X_n\}$ be a set of PCA. A pseudo execution fragment of $X$ is a pseudo execution fragment of $\text{psioa}(X)$, s. t. for every non final state $q'$, $X$ is partially-compatible at state $q'$ (namely the conditions (1) and (3) need to be satisfied)

A pseudo execution $e$ of $X$ is a pseudo execution fragment of $X$ with $f\text{state}(e) = (q_{X_1}, ..., q_{X_n})$.

**Definition 5.21 (reachable state).** Let $X = \{X_1, ..., X_n\}$. A state $q$ of $X$ is reachable if it exists a pseudo execution $e$ of $X$ ending on state $q$.

Now, we are able to define our composition operator.

**Definition 5.22 (Composition of configuration automata).** Let $X_1, ..., X_n$, partially-compatible PCA. Then $X = X_1||...||X_n$ is the state machine consisting of the following components:

1. $\text{psioa}(X) = \text{psioa}(X_1)||...||\text{psioa}(X_n)$
2. A configuration mapping $\text{config}(X)$ given as follows. For each $x = (x_1, ..., x_n) \in \text{states}(X), \text{config}(X)(x) = \text{config}(X_1)(x_1) \cup ... \cup \text{config}(X_n)(x_n)$
3. For each $x = (x_1, ..., x_n) \in \text{states}(X)$, a mapping $\text{created}(X)(x)$ with domain $\tilde{\sigma}(X)(x)$ and given as follows. For each $a \in \tilde{\sigma}(X)(x)$, $\text{created}(X)(x)(a) = \bigcup_{\tilde{\sigma}(X_i)(x_i) \in \text{states}(X)} \text{created}(X_i)(x_i)(a)$
4. A hidden-action mapping $\text{hidden-actions}(X)$ with domain $\text{states}(X)$ and given as follows. For each $x = (x_1, ..., x_n) \in \text{states}(X)$, $\text{hidden-actions}(x) = \bigcup_{x_i \in \text{states}(X)} \text{hidden-actions}(x_i)$

We define $\text{states}(X) = \text{states}(\text{psioa}(X)), \text{start}(X) = \text{start}(\text{psioa}(X)), \tilde{\sigma}(X) = \tilde{\sigma}(\text{psioa}(X)), \text{steps}(X) = \text{steps}(\text{psioa}(X))$, and likewise for all other (sub)components and attributes of $\text{psioa}(X)$.

We want to show that the set of PCA is closed under composition. Before starting the proof, we introduce some tools.

**Lemma 5.23 (Joint preserving probability distribution for union of configuration).** Let $\{C_k = (A_k, S_k)\}_{k \in [1:n]}$ be a finite set of compatible configurations s. t. $\forall k, \ell \in [1:n], A_k \cap A_\ell = \emptyset$. Let $C = (A, S) = \bigcup_{k \in [1:n]} C_k$ be a compatible configuration. Let $a \in \tilde{\sigma}(C)$. Let $(I, J)$ be a partition of $[1:n]$ s. t. for every $i \in I$, $a \in \tilde{\sigma}(C_i)$ and for every $j \in J$, $a \in \tilde{\sigma}(C_j)$. For every $i \in I$, let $\eta^i_p$ be the unique preserving distributions that has $\eta(A_k, q_i, a)$ as companion distribution with $q_i = US(C_i)$. For every $j \in J$, let $\eta^j_p = \delta_{C_j}$. We note $\eta_p$ the unique preserving distributions that have $\eta(A, q, a)$ as companion distribution with $q = US(C)$.

Then, $C \triangleq \eta_p$, s. t. for every configuration $C' = (A, S')$, for every (unique) finite set of configurations $\{C'_k = (A_k, S'_k)\}_{k \in [1:n]}$ verifying $C' = \bigcup_{k \in [1:n]} C'_k$, we have $\eta_p(C') = (\eta^1_p \otimes ... \otimes \eta^n_p)(\bigcup_{k \in [1:n]} C'_k)$.

**Proof.** Since $A = \bigcup_{k \in [1:n]} A_k$ and $S$ agrees with $S_k$ on $A \in A_k$ for every $k \in [1:n]$, we have $\eta(A, q, a) = \eta(A_k, q_k, a) \otimes ... \otimes \eta(A_n, q_n, a)$ with the convention $\eta(A_{\emptyset}, q_{\emptyset}, a) = \delta_{q_{\emptyset}}$ for every $j \in J$. Furthermore, for every $k \in [1:n], \eta(A_k, q_k, a)$ is a companion distribution of $\eta_p^k$, that is for every $C'_k, q'_k = US(C'_k), \eta^k_p(C'_k) = \eta(A_k, q_k, a)$. Hence for every $(C'_1, ..., C'_n)$, $(q'_1 = US(C'_1), ..., q'_n = US(C'_n))$, $\eta^k_p(C'_k) = (\eta(A_k, q_k, a))((q'_1, ..., q'_n)) = (\eta^1_p \otimes ... \otimes \eta^n_p)((C'_1, ..., C'_n))$ (1). By definition of $\eta_p$, for every $C'$, $(q' = US(C'))$, $\eta(A, q, a)(q') = \eta_p(C')$. Since we deal with preserving distribution and $A = \bigcup_{k \in [1:n]} A_k$, we have each element $q' = US(C') \in QA$ is of the form $(q'_1, ..., q'_n)$ with $q'_k \in Q_{A_k}$ and verifies $C' = C_1 \cup ... \cup C_n$ with $auto(C'_k) = A_k$ and $US(C'_k) = q'_k$. Hence we compose (*) and (**) to obtain for every configuration $C' = (A, S')$, for every finite set of configurations $\{C'_k = (A_k, S'_k)\}_{k \in [1:n]}$ s. t. $C' = \bigcup_{k \in [1:n]} C'_k$, then...
\[\eta_p(C') = (\eta_1^p \otimes \ldots \otimes \eta_n^p)((C_1', \ldots, C_n'))\] Since \(a \in \text{sig}(C)\) and the automata support of \((\eta_1^p \otimes \ldots \otimes \eta_n^p)\) is \(A = \bigcup_{k \in [1,n]} A_k\), we can note \(C \xrightarrow{a} \eta_p\)

**Definition 5.24 (merge, join).** Let \(\eta = (\eta_1, \ldots, \eta_n) \in \text{Disc}(\text{Conf}_p)^n\). We define

- \(\text{join}(\eta) : (C_1, \ldots, C_n) \in Q^n_{\text{Conf}} \mapsto (\eta_1 \otimes \ldots \otimes \eta_n)(C_1, \ldots, C_n)\) and
- \(\text{merge}(\eta) : C \in \text{Conf} \mapsto \sum_{(c_1, \ldots, c_n) \in Q^n_{\text{Conf}}} \text{join}(\eta)((c_1^k, \ldots, c_n^k)) \cdot \varphi((c_1^k \cup \ldots \cup c_n^k) = C)\)

**Definition 5.25 (deter-dest, base).** Let \(C^3 = (A^3, S^3)\) be a configuration. For every \(A \in A^3\), we note \(q^3_A = S^3(A)\). Let \(\varphi \in \mathcal{P}(\text{Autids})\). We define

- \(\text{deter-dest}(C^3, a) = \{A \in A^3 | a \in \text{sig}(A)(q^3_A) \land \eta_A.q^3_A.a = \delta_{q^3_A} \}\)
- \(\text{base}(C^3, a, \varphi) = A^3 \cup \varphi \setminus \text{deter-dest}(C^3, a)\)

**Lemma 5.26 (Merging).** Let \((C_1^3, \ldots, C_n^3)\) be a set of compatible configuration with for every \(i \in [1,n], C_i^3 = (A_i^3, S_i^3)\). Let \(C^3 = (A^3, S^3) = \bigcup_{i \in [1,n]} C_i^3\). For every \(A \in A^3\), we note \(q^3_A = S^3(A)\). Let \(a \in \text{sig}(C^3)\). Let \((I, J)\) be a partition of \([1,n]\) s.t. \(\forall i \in I, a \in \text{sig}(C_i^3)\) and \(\forall j \in J, a \notin \text{sig}(C_j^3)\). Let \((\varphi_1, \ldots, \varphi_n) \in \mathcal{P}(\text{Autids})^n\), s.t. \(\forall j \in J, \varphi_j = 0\) and let us note \(\varphi = \bigcup_{i \in I} \varphi_i\). For every \(i \in I\), we note \(\eta_i s.t. C_i^3 \xrightarrow{a} \eta_i\). For every \(j \in J\), we note \(\eta_j s.t. \eta_j = \delta_{C_j^3}\).

Let \(\tilde{\eta} = (\eta_1, \ldots, \eta_n)\). Let \(\eta\text{join} = \text{join}(\tilde{\eta})\), \(\eta\text{merge} = \text{merge}(\tilde{\eta})\) and let assume, for every \(C^f \in \text{supp}(\eta\text{merge})\), either \((*)\) \(C^f\) is compatible or \((**): \forall k \in [1,n], \text{base}(C_k^f, a, \varphi_k) \cap \text{base}(C_k^3, a, \varphi_k) = 0\). Then, it means

1. \(\forall C^f \in \text{supp}(\eta\text{merge}), \text{it exists a unique } (C_1^f, \ldots, C_n^f) \text{ s.t. } 1) C^f = C_1^f \cup \ldots \cup C_n^f \text{ and 2) } \forall k \in [1,n], C_k^f \in \text{supp}(\eta_k)\).

We note \(C^f\text{split}(\tilde{\eta})\) this unique \((C_1^f, \ldots, C_n^f)\).

2. \(g^f : C \in \text{supp}(\text{merge}(\tilde{\eta})) \mapsto C\text{split}(\tilde{\eta}) \in \text{supp}(\eta_1) \times \ldots \times \text{supp}(\eta_n)\) is a bijection.

3. Then \(\text{merge}(\tilde{\eta})(C) = \text{join}(\tilde{\eta})(C\text{split}(\tilde{\eta}))\).

(4) \(merge(\tilde{\eta}) \xrightarrow{a} \text{join}(\tilde{\eta})\)

(5) \(C^3 \xrightarrow{a} \text{merge}(\tilde{\eta})\)

**Proof.** First we show that \((*)\) implies \((**).\) By contradiction. Let assume \(A \in \text{base}(C_k^f, a, \varphi_k) \cap \text{base}(C_k^3, a, \varphi_k)\). Since \(C^3\) is compatible, \(A \notin A_k^3 \cup A_k^f\). By definition of the base it exists \(C_k^f, C_k^3 \in \text{supp}(\eta_k) \times \text{supp}(\eta_k), A \in \text{auts}(C_k^f) \cap \text{auts}(C_k^3)\) and \(C_k^f \cup C_k^3\) is not compatible. So it exists \((C_1^f, \ldots, C_n^f) \in \text{supp}(\eta_1 \otimes \ldots \otimes \eta_n)\) s.t. \((C_1^f \cup \ldots \cup C_n^f)\) is not compatible.

1. The uniqueness comes from \((**)\). Indeed, let imagine two candidates \((C_1^f, \ldots, C_n^f)\) and \((C_1^f', \ldots, C_n^f')\) if \(A \in \text{auts}(C_k^f)\), then \(A \in \text{auts}(C_k^3)\) but \(A \notin \text{base}(C_k^f)\) for any \(k' \neq k\) and so \(A \in \text{auts}(C_k^f)\) and so \(\text{auts}(C_k^f) = \text{auts}(C_k^3)\) for every \(k \in [1,n]\). Now we need to have \(\text{map}(C_k^f) = \text{map}(C_k^f')\) for every \(k \in [1,n]\) to obtain \(C^f = C_1^f \cup \ldots \cup C_n^f = C_1^f \cup \ldots \cup C_n^f\). The existence is by construction of join.

2. The bijection comes from the existence and the uniqueness of pre-image

3. By previous item 1, the sum of merge has only one element which consists of \(\text{join}(\tilde{\eta})(C\text{split}(\tilde{\eta}))\)

4. By previous two items 1 and 2, that gives the definition

(5) \(\eta_k^p\) is generated by \(\varphi_k\) and preserving distribution \(\eta_k^p\) where \(A_k^p\) is the automata support of \(\eta_k^p\). By compatibility of \(C^3\), for every \(k, t \in [1,n], k \neq t\), \(A_k^p \cap A_t^p = \emptyset\). Hence, we can apply lemma 5.23 and we have \(C^3 \xrightarrow{a} \text{merge}((\eta_1^p, \ldots, \eta_n^p)).\) Moreover \(\text{merge}((\eta_1', \ldots, \eta_n'))\) is generated by \(\text{merge}((\eta_1^p, \ldots, \eta_n^p))\) and \(\varphi\) since each \(\eta_k^p\) is generated by \(\eta_k^p\) and \(\varphi_k\). Then \(C^3 \xrightarrow{a} \text{merge}((\eta_1', \ldots, \eta_n'))\)
LEMMA 5.27 (f→ preservation for joint probability). Let $\eta' = (\eta_1', ..., \eta_n')$. Let $(\eta_1, ..., \eta_n)$ and $(f_1, ..., f_n)$ s. t. $\forall i \in [1, n], \eta_i \xrightarrow{f_i} \eta_i'$.

Then $\eta_1 \otimes ... \otimes \eta_n \xrightarrow{\eta_1' \otimes ... \otimes \eta_n'}$. With $f : x = (x_1, ..., x_n) \mapsto y = (f_1(x_1), ..., f_n(x_n))$.

Proof. $f$ is still a bijection and $(\eta_1 \otimes ... \otimes \eta_n)(z_1, ..., z_n) = \eta_1(z_1) \cdot ... \cdot \eta_n(z_n) = \eta_1'(f_1(z_1)) \cdot ... \cdot \eta_n'(f_n(z_n)) = (\eta_1 \otimes ... \otimes \eta_n)(f(z_1, ..., z_n))$.

Now we are ready for the theorem that claims that a composition of PCA is a PCA.

THEOREM 5.28 (PCA Closeness under composition). Let $X_1, ..., X_n$, be partially-compatible PCA. Then $X = X_1[...][X_n]$ is a PCA.

Proof. We need to show that $X$ verifies all the constraints of definition 5.14.

- (Constraint 1) The demonstration is basically the same as the one in [1], section 5.1, proposition 21, p 32-33.
  Let $\bar{q}_X$ and $(\bar{A}, \bar{S}) = config(X)(\bar{q}_X)$. By the composition of psiao, then $\bar{q}_X = (\bar{q}_X_1, ..., \bar{q}_X_n)$. By definition, config$(X)(\bar{q}_X) = config(X_1)(\bar{q}_X_1) \cup ... \cup config(X_n)(\bar{q}_X_n)$. Since for every $j \in [1 : n], X_j$ is a configuration automaton, we apply constraint 1 to $X_j$ to conclude $S(A_{\bar{q}_j}) = \bar{q}_{X_j}$ for every $A_{\bar{q}_j} \in a(A(config(X_1)(\bar{q}_X_1))$. Since $(a(config(X_1)(\bar{q}_X_1), ..., a(config(X_n)(\bar{q}_X_n))$ is a partition of $A$ by definition of composition, $S(A_{\bar{q}_j}) = \bar{q}_{X_j}$ for every $A_{\bar{q}_j} \in A$ which ensures $X$ verifies constraint 1.

- (Constraint 2) Let $(q, a, \eta_{X,q,a}) \in dtrans(X)$. We will establish $\exists \eta' \in Disc(Q_{con'}(q)) s. t. \eta_{X,q,a} \xrightarrow{f} \eta'$ where $f = config(X)$ and config$(X)(q) \xrightarrow{a} \eta'$ with $\varphi = created(X)(q)(a)$.

For brevity, let $\bar{A}_i = psiao(X_i)$ for $i \in [1 : n]$. By definition of pca-composition 5.22, $psiao(X) = psiao(X_1)[..][psiao(X_n) = P_1[...][P_n$. We note $P = (P_1, ..., P_n)$ and then by definition of psiao-composition 4.10, $q = (q_1, ..., q_n) \in QP_1 \times ... \times QP_n$, while $a \in \bigcup_{i \in [1:n]} sig(P_i)(q_i)$ and $\eta_{X,q,a} = \eta_{P_1,q_1,a} \otimes ... \otimes \eta_{P_n,q_n,a}$ with the convention $\eta_{P_i,q_i,a} = \delta_{q_i}$ if $a \notin sig(P_i)(q_i)$.

Let $(I, J)$ be a partition $[1 : n]$ s. t.

For every $i \in I$, $a \in sig(P_i)(q_i)$, and then by PCA top/down transition preservation, it exists $\eta' \in Disc(Q_{con'}) s. t. \eta_{X,q,a} \xrightarrow{f} \eta'$ with $f_i = config(X_i)$ and config$(X_i)(q_i) \xrightarrow{a}_{\varphi_i} \eta_i'$ with $\varphi_i = created(X_i)(q_i)(a)$.

For every $j \in J$, $a \notin sig(P_i)(q_i)$, then we note $\varphi_j = \emptyset$ and $\eta'_j = \delta_{config(X_j)}(q_j)$ that verifies $\delta_{\eta'_j} \xrightarrow{\eta_i'_{J}} \eta'$ with $f_j = config(X_j)$.

We note $\eta' = (\eta_1', ..., \eta_n')$. We note $\eta' = join(\eta') = \eta_1' \otimes ... \otimes \eta_n'$ and $\varphi = \bigcup_{i \in [1:n]} \varphi_i$. By definition of PCA-composition, $\varphi = created(X)(q)(a)$.

We have $\eta_{X,q,a} \xrightarrow{f} \eta'$ with $f : q = (q_1, ..., q_n) \mapsto (f_1(q_1), ..., f_n(q_n))$ by lemma 5.27.

Moreover merge$(\tilde{\eta}') \xrightarrow{\tilde{\eta}} join(\tilde{\eta}')$ with $\tilde{\eta} : C \in supp(merge(\tilde{\eta}))(\mapsto C.split(\tilde{\eta}) \in supp(\eta_1) \times ... \times supp(\eta_n)$ by lemma 5.26, item 4.

So $\eta_{X,q,a} \xrightarrow{h} merge(\tilde{\eta}')$ with $h = (\tilde{\eta})^{-1} \circ f = config(X)$.

Moreover we have config$(X)(q) \xrightarrow{a} merge(\tilde{\eta}')$ by lemma 5.26, item 5.

- (Constraint 3) Let $q \in state(X), C = config(X)(q), a \in sig(X)(q), \varphi = created(X)(q)(a)$ that verify $C \xrightarrow{\varphi} \eta'$.

We need to show that it exists $(q, a, \eta_{X,q,a}) \in dtrans(X) s. t. \eta_{X,q,a} \xrightarrow{f} \eta'$ with $f = config(X)$.
For brevity, let $A_i = \text{psioa}(X_i)$ for $i \in [1 : n]$. By definition of pca-composition 5.22 $\text{psioa}(X) = \text{psioa}(X_1) \mid ... \mid \text{psioa}(X_n) = P_1 \mid ... \mid P_n$. We note $P = (P_1, ..., P_n)$ and then by definition of psioa-composition, $q = (q_1, ..., q_n) \in Q_1 \times ... \times Q_n$, while $a \in \bigcup_{i \in [1 : n]} \widehat{\text{sig}}(P_i(q_i))$.

Let $(I, J)$ be a partition $[1 : n]$ s.t.

For every $i \in I$, $a \in \text{sig}(P_i(q_i))$. We note $\varphi_i = \text{created}(X_i)(q_i)$. For every $j \in J$, $a \notin \text{sig}(P_j(q_j))$, then we note $\varphi_j = 0$ and $\eta'_j = \delta_{\text{config}(X_j)(q_j)}$ that verifies $\delta_{\eta_j} f_j \leftrightarrow \eta'_j$ with $f_j = \text{config}(X_j)$.

We note $\varphi = \text{created}(X)(q)(a)$. By pca-composition definition, $\varphi = \bigcup_{k \in [1 : n]} \eta_k$. For every $k \in [1 : n]$, we note $C_k = \text{config}(X_k)(q_k)$ and for every $i \in I$, $\eta'_i \in \text{Disc}(Q_{\text{conf}})$ s.t. $C_i \xrightarrow{a} \varphi_i \eta'_i$. We note $\eta' = (\eta'_1, ..., \eta'_{n})$

By constraint 3 (bottom-up transition preservation), for every $i \in I$, it exists $(q_i, a, \eta_{X_i, q_i, a}) \in \text{dtrans}(X_i)$ s.t. $\eta_{X_i, q_i, a} \overset{h_i}{\leftrightarrow} \eta'_i$ with $f_i = \text{config}(X_i)$, by lemma 5.27, $\eta_{X, q, a} = \eta_{X_1, q_1, a} \otimes ... \otimes \eta_{X_n, q_n, a} \overset{h}{\leftrightarrow} \eta'_1 \otimes ... \otimes \eta'_{n}$ = $\text{join}(\eta')$ with the convention $\eta_{X_i, q_i, a} = \delta_{q_i}$, for $j \in J$ and $f : q = (q_1, ..., q_n) \in \text{states}(X) \mapsto (f_1(q_1), ..., f_n(q_n))$.

By partially-compatibility, for every $C^f \subseteq \text{supp}(\text{merge}(\eta'))$, $C^f$ is compatible. Hence we can apply lemma 5.26, item 5, which gives $\text{merge}(\eta') \overset{h}{\leftrightarrow} \text{join}(\eta')$ with $g : C \in \text{supp}(\text{merge}(\eta')) \mapsto C.\text{split}(\eta)$.

Hence $\eta_{X, q, a} \overset{h}{\leftrightarrow} \text{merge}(\eta')$ with $h = g^{-1} \circ f$, that is $\eta_{X, q, a} \overset{h}{\leftrightarrow} \eta'$ with $h' = \text{config}(X)$ and the restriction of $h'$ on $\text{supp}(\eta_{X, q, a})$ is $h$.

(Constraint 4).

For every $i \in [1 : n]$, we note $h_X = \text{hidden-actions}(X_i)(q_{X_i})$ and $h = \bigcup_{i \in [1 : n]} h_X$. Since $X_1, ..., X_n$ are partially-compatibly in state $q_X = (q_{X_1}, ..., q_{X_n})$, we have both $(\text{config}(X_i)(q_{X_i})) \mid i \in [1, n]$ compatible and $\forall i, j \in [1, n], (\text{in}(\text{config}(X_i)(q_{X_i})) \cap h_X = \emptyset$. By compatibility, $\forall i, j \in [1, n], i \neq j, \text{out}(\text{config}(X_i)(q_{X_i})) \cap \text{out}(\text{config}(X_j)(q_{X_j})) = \emptyset$, which finally gives $\forall i, j \in [1, n], i \neq j, \text{sup}(\text{config}(X_i)(q_{X_i})) \cap h_X = \emptyset$.

Hence, we can apply lemma 4.25 of commutativity between hiding and composition to obtain $\text{hide}(\text{sig}(\text{config}(X_1)(q_{X_1})) \times ... \times \text{sig}(\text{config}(X_n)(q_{X_n})), h_{X_1} \cup ... \cup h_{X_n} = \text{hide}(\text{sig}(\text{config}(X_1)(q_{X_1})), h_{X_1}) \times ... \times \text{hide}(\text{sig}(\text{config}(X_n)(q_{X_n})), h_{X_n})$ where $\times$ has to be understood in sense of definition 4.5 of signature composition.

That is $\text{sig}(\text{psioa}(X))(q_X) = \text{sig}(\text{psioa}(X_1))(q_{X_1}) \times ... \times \text{sig}(\text{psioa}(X_n))(q_{X_n})$, as per definition 4.5, with $\text{sig}(\text{psioa}(X))(q_X) = \text{hide}(\text{sig}(\text{config}(X)(x)), h)$. Furthermore $h \in \text{out}(\text{config}(X)(q_X))$, since $\forall i \in [1, n], h_X \subset \text{out}(\text{config}(X_i)(q_{X_i}))$. This terminates the proof.

\[\square\]

### 6 INTRODUCTION ON PCA CORRESPONDING W.R.T. PSIOA $\mathcal{A}, \mathcal{B}$ TO INTRODUCE MONOTONICITY

In this section we take an interest in PCA $X_\mathcal{A}$ and $X_\mathcal{B}$ that differ only on the fact that $\mathcal{B}$ supplants $\mathcal{A}$ in $X_\mathcal{B}$. This definition is a key step to formally define monotonicity of a property.

If a property $P$ is a binary relation on automata, a brave property would verify monotonicity, i.e. if 1) $(\mathcal{A}, \mathcal{B}) \in P$, and 2) $X_\mathcal{A}$ and $X_\mathcal{B}$ are PCA that differ only on the fact that $\mathcal{B}$ supplants $\mathcal{A}$ in $X_\mathcal{B}$, then 3) $(X_\mathcal{A}, X_\mathcal{B}) \in P$. Monotonicity of implementation w.r.t. PSIOA creation is the main contribution of the paper.

#### 6.1 Naive correspondence between two PCA

We formalize the idea that two configurations are identical except that the automaton $\mathcal{B}$ supplants $\mathcal{A}$ but with the same external signature. The following definition comes from [1].
**Definition 6.1 (≺_{\mathcal{AB}}-corresponding configurations).** (see figure 30) Let $\Phi \subseteq \text{Autids}$, and $\mathcal{A}, \mathcal{B}$ be PSIOA identifiers. Then we define $\Phi[\mathcal{B}/\mathcal{A}] = (\Phi \setminus \mathcal{A}) \cup \{\mathcal{B}\}$ if $\mathcal{A} \not\in \Phi$, and $\Phi[\mathcal{B}/\mathcal{A}] = \Phi$ if $\mathcal{A} \not\in \Phi$. Let $C, D$ be configurations. We define $C \prec_{\mathcal{AB}} D$ iff (1) $\text{auts}(D) = \text{auts}(C)[\mathcal{B}/\mathcal{A}]$, (2) for every $\mathcal{A}' \not\in \text{aut}(C) \setminus \{\mathcal{A}\} : \text{map}(D)(\mathcal{A}') = \text{map}(C)(\mathcal{A}')$, and (3) $\text{ext}(\mathcal{A})(s) = \text{ext}(\mathcal{B})(t)$ where $s = \text{map}(C)(\mathcal{A}), t = \text{map}(D)(\mathcal{B})$. That is, in $\prec_{\mathcal{AB}}$-corresponding configurations, the SIOA other than $\mathcal{A}, \mathcal{B}$ must be the same, and must be in the same state. $\mathcal{A}$ and $\mathcal{B}$ must have the same external signature. In the sequel, when we write $\Psi = \Phi[\mathcal{B}/\mathcal{A}]$, we always assume that $\mathcal{B} \not\in \Phi$ and $\mathcal{A} \not\in \Psi$.

![Fig. 17. $\prec_{\mathcal{AB}}$ corresponding-configuration](image)

**Remark 2.** It is possible to have two configurations $C, D$ s.t. $C \prec_{\mathcal{AB}} D$. That would mean that $C$ and $D$ only differ on the state of $\mathcal{A}$ (or $\mathcal{B}$) that has even the same external signature in both cases $\text{ext}(\mathcal{A})(s)$ is $\text{ext}(\mathcal{A})(t)$, while we would have $\text{int}(\mathcal{A})(s) \not= \text{int}(\mathcal{A})(t)$.

Now, we formalise the fact that two PCA create some PSIOA in the same manner, excepting for $\mathcal{B}$ that supplants $\mathcal{A}$. Here again, this definition comes from [1].

**Definition 6.2 (Creation corresponding configuration automata).** Let $X, Y$ be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that $X, Y$ are creation-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff

1. $X$ never creates $\mathcal{B}$ and $Y$ never creates $\mathcal{A}$.
2. $\beta \in \text{traces}^*(X) \cap \text{traces}^*(Y)$ a finite trace of both $X$ and $Y$, and let $\alpha \in \text{Exec}^*(X), \pi \in \text{Exec}^*(Y)$ a finite execution of both $X$ and $Y$ be such that $\text{trace}_{\mathcal{A}}(\alpha) = \text{trace}_{\mathcal{B}}(\pi) = \beta$. Let $x = \text{last}(\alpha), y = \text{last}(\pi)$, i.e., $x, y$ are the last states along $\alpha, \pi$, respectively. Then $\forall a \in \text{sig}(X)(x) \cap \text{sig}(Y)(y) : \text{created}(Y)(y)(a) = \text{created}(X)(x)(a)[\mathcal{B}/\mathcal{A}]$.

In the same way than in definition 6.2, we formalise the fact that two PCA hide some output actions in the same manner. Here again, this definition is inspired by [1].

**Definition 6.3 (Hiding corresponding configuration automata).** Let $X, Y$ be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that $X, Y$ are hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff

1. $X$ never creates $\mathcal{B}$ and $Y$ never creates $\mathcal{A}$.
2. $\beta \in \text{traces}^*(X) \cap \text{traces}^*(Y)$, and let $\alpha \in \text{Exec}^*(X), \pi \in \text{Exec}^*(Y)$ be such that $\text{trace}_{\mathcal{A}}(\alpha) = \text{trace}_{\mathcal{B}}(\pi) = \beta$. Let $x = \text{last}(\alpha), y = \text{last}(\pi)$, i.e., $x, y$ are the last states along $\alpha, \pi$, respectively. Then $\text{hidden-actions}(Y)(y) = \text{hidden-actions}(X)(x)$.
Now we define the notion of \(\mathcal{A}\)-exclusive action which corresponds to an action which is is the signature of \(\mathcal{A}\) only. This definition is motivated by the fact that monotonicity induces that \(\mathcal{A}\)-exclusive (resp. \(\mathcal{B}\)-exclusive) actions do not create automata. Indeed, otherwise two internal action \(a\) and \(a'\) of \(\mathcal{A}\) and \(\mathcal{B}\) respectively could create different automata \(C\) and \(D\) and break the correspondence.

**Definition 6.4 (\(\mathcal{A}\)-exclusive action).** Let \(\mathcal{A} \in \text{Autids}, X\) be a PCA. Let \(q \in \text{states}(X), (A, S) = \text{conf}(X)(q), \text{act} \in \text{sig}(X)(q)\). We say that \(\text{act}\) is \(\mathcal{A}\)-exclusive if for every \(\mathcal{A}' \in \mathcal{A} \setminus \{\mathcal{A}\}, \text{act} \notin \text{sig}(\mathcal{A}')(\text{S}(\mathcal{A}'))\) (and so \(\text{act} \in \text{sig}(\mathcal{A})(\text{S}(\mathcal{A}))\) only).

The previous definitions 6.1, 6.2, 6.3 and 6.4 allow us to define a first (naive) definition of PCA corresponding w. r. t. \(\mathcal{A}, \mathcal{B}\).

**Definition 6.5 (naively corresponding w. r. t. \(\mathcal{A}, \mathcal{B}\)).** Let \(\mathcal{A}, \mathcal{B} \in \text{Autids}, X_A, X_B\) be PCA we say that \(X_A\) and \(X_B\) are naively corresponding w. r. t. \(\mathcal{A}, \mathcal{B}\), if they verify:
- \(\text{conf}(X_A)(\bar{q}_{X_A}) \prec_{\mathcal{A}, \mathcal{B}} \text{conf}(X_B)(\bar{q}_{X_B})\).
- \(X_A, X_B\) are creation-corresponding w.r.t. \(\mathcal{A}, \mathcal{B}\)
- \(X_A, X_B\) are hiding-corresponding w.r.t. \(\mathcal{A}, \mathcal{B}\)
- (No creation from \(\mathcal{A}\) and \(\mathcal{B}\))
  - \(\forall q_{X_A} \in \text{states}(X_A)\), for every action \(\text{act} \mathcal{A}\)-exclusive, \(\text{created}(X_A)(q_{X_A})(\text{act}) = \emptyset\) and similarly
  - \(\forall q_{X_B} \in \text{states}(X_B)\), for every action \(\text{act}' \mathcal{B}\)-exclusive, \(\text{created}(X_B)(q_{X_B})(\text{act}') = \emptyset\)

The last definition 6.5 of (naive) correspondence w. r. t. \(\mathcal{A}, \mathcal{B}\) allows us to define a first (naive) definition 6.6 of monotonic relation.

**Definition 6.6 (Naively monotonic relationship).** Let \(R\) be a binary relation on PSIOA. We say that \(R\) is naively monotonic if for every pair of PSIOA \((\mathcal{A}, \mathcal{B}) \in R\), for every pair of PCA \(X_A, X_B\) that are naively corresponding w. r. t. \(\mathcal{A}, \mathcal{B}\) \((\text{psioa}(X_A), \text{psioa}(X_B)) \in R\).

However, the relations of print-implementation introduced in subsection 4.6 is not monotonic without some additional technical assumptions presented in next subsection 6.2. Roughly speaking, it allows to 1) define a PCA \(Y = X \setminus \{\mathcal{A}\}\) that corresponds to \(X\) 'deprived' from \(\mathcal{A}\) and 2) define the composition between \(Y\) and \(\mathcal{A}\), 3) avoiding some ambiguities during the construction. In the first instance, the reader should skip the next subsection 6.2 on conservatism and keep in mind the intuition only. This sub-section 6.2 can be used to know the assumptions of the theorems of monotonicity and use them as black-boxes. The assumptions will be re-called during the proof.

### 6.2 Conservatism: the additional assumption for relevant definition of correspondence w. r. t. \(\mathcal{A}, \mathcal{B}\)

This subsection aims to define the notion of \(\mathcal{A}\)-conservative PCA.

**Some definitions relative to configurations.** In the remaining, it will often be useful to reason on the configurations. This is why we introduce some definitions that will be used again and again in the demonstrations.

The next definition captures the idea that two states of a certain layer represents the same situation for the bottom layer.
Definition 6.7 (configuration-equivalence between two states). Let \( K, K' \) be PCA and \((q, q') \in \text{states}(K) \times \text{states}(K')\). We say that \( q \) and \( q' \) are config-equivalent, noted \( qR_{\text{conf}}q' \), if \( \text{config}(K)(q) = \text{config}(K')(q') \). Furthermore, if

- \( \text{config}(K)(q) = \text{config}(K')(q') \),
- \( \text{hidden-actions}(K)(q) = \text{hidden-actions}(K')(q') \) and
- \( \forall a \in \text{sig}(K)(q) = \text{sig}(K')(q') \), \( \text{created}(K)(q)(a) = \text{created}(K')(q')(a) \),

we say that \( q \) and \( q' \) are strictly-equivalent, noted \( qR_{\text{strict}}q' \).

Now, we define a special subset of PCA that do not tolerate different configuration-equivalent states.

Definition 6.8 (Configuration-conflict-free PCA). Let \( K \) be a PCA. We say \( K \) is configuration-conflict-free, if for every \( q, q' \in \text{states}(K) \) s. t. \( qR_{\text{conf}}q' \), then \( q = q' \). The current state of a configuration-conflict-free PCA can be defined by its current attached configuration.

For some elaborate definitions, we found useful to introduce the set of potential output actions of \( \mathcal{A} \) in a configuration \( \text{config}(X)(q) \) coming from a state \( q \) of a PCA \( X \):

Definition 6.9 (potential output). Let \( \mathcal{A} \in \text{autids}. \) Let \( X \) be a PCA. Let \( q \in \text{states}(X) \). We note \( \text{pot-out}(X)(q)(\mathcal{A}) \) the set of potential output actions of \( \mathcal{A} \) in \( \text{config}(X)(q) \) that is

- \( \text{pot-out}(X)(q)(\mathcal{A}) = \emptyset \) if \( \mathcal{A} \not\in \text{auts}(\text{config}(X)(q)) \)
- \( \text{pot-out}(X)(q)(\mathcal{A}) = \text{out}(\mathcal{A})(\text{map}(\text{config}(X)(q))(\mathcal{A})) \) if \( \mathcal{A} \in \text{auts}(\text{config}(X)(q)) \)

Here, we define a configuration \( C \) deprived from an automaton \( \mathcal{A} \) in the most natural way.

Definition 6.10 (\( C \setminus \{ \mathcal{A} \} \) Configuration deprived from an automaton). \( C = (A, S), C \setminus \{ \mathcal{A} \} = (A', S') \) with \( A' = A \setminus \{ \mathcal{A} \} \) and \( S' \) the restriction of \( S \) on \( A' \).

The two last definitions 6.9 and 6.10 allows us to define in compact way a new relation between states that captures the idea that two states \( q \in \text{states}(X) \) and \( q' \in \text{states}(Y) \) are equivalent modulo a difference uniquely due to the presence of automaton \( \mathcal{A} \) in \( \text{config}(X)(q) \) and \( \text{config}(Y)(q') \).

Definition 6.11 (\( R_{\text{conf}}^{(\setminus \{ \mathcal{A} \})} \) relationship (equivalent if we forget \( \mathcal{A} \)). Let \( \mathcal{A} \in \text{Autids}. \) Let \( S = \{\text{states}(X)|X \text{ is a PCA}\} \) the set of states of any PCA. We defined the equivalence relation \( R_{\text{conf}}^{(\setminus \{ \mathcal{A} \})} \) on \( S \) defined by \( \forall X, Y \) PCA, \( \forall (q_X, q_Y) \in \text{states}(X) \times \text{states}(Y) \):

- \( q_X R_{\text{conf}}^{(\setminus \{ \mathcal{A} \})} q_Y \iff \text{config}(X)(q_X) \setminus \{ \mathcal{A} \} = \text{config}(Y)(q_Y) \setminus \{ \mathcal{A} \} \)
- \( q_X R_{\text{strict}}^{(\setminus \{ \mathcal{A} \})} q_Y \iff \) the conjunction of the 3 following properties:
  - \( \forall a \in \text{sig}(X)(q_X) \cap \text{sig}(Y)(q_Y), \text{created}(Y)(q_Y)(a) \setminus \{ \mathcal{A} \} = \text{created}(X)(q_X)(a) \setminus \{ \mathcal{A} \} \)
  - \( \text{hidden-actions}(X)(q_X) \setminus \text{pot-out}(X)(q_X)(\mathcal{A}) = \text{hidden-actions}(Y)(q_Y) \setminus \text{pot-out}(Y)(q_Y)(\mathcal{A}) \)

Here, we recall the definition 6.4 of \( \mathcal{A} \)-exclusive action:

Definition 6.12 (\( \mathcal{A} \)-exclusive action in a PCA state (recall)). Let \( \mathcal{A} \in \text{autids}. \) Let \( X \) be a PCA. Let \( q_X \in \text{states}(X), (A, S) = \text{config}(X)(q_X), a \in \text{sig}(X)(q_X) \).

We say that \( a \) is \( \mathcal{A} \)-exclusive in \( q_X \) if \( \forall B \in A \setminus \{ \mathcal{A} \}, a \notin \text{sig}(B)(S(B)) \) (and so \( a \in \text{sig}(\mathcal{A})(S(\mathcal{A})) \) uniquely necessarily).
\[ \mathcal{A}\text{-}fair \text{ and } \mathcal{A}\text{-}conservative: \text{ necessary assumptions to authorize the construction used in the proof.} \text{ Now, we are ready to define } \mathcal{A}\text{-}fairness \text{ and then } \mathcal{A}\text{-}conservatism. \]

A \( \mathcal{A}\text{-}fair \) PCA is a PCA s. t. we can deduce its current properties from its current configuration deprived of \( \mathcal{A} \). This assumption will allow us to define \( Y = X \setminus \{ \mathcal{A} \} \) in the proof of monotonicity.

**Definition 6.13 (\( \mathcal{A}\text{-}fair \) PCA).** Let \( \mathcal{A} \in \autids \). Let \( X \) be a PCA. We say that \( X \) is \( \mathcal{A}\text{-}fair \) if

- (configuration-conflict-free) \( X \) is configuration-conflict-free.
- (no conflict for projection) \( \forall q_X, q_X' \in \text{states}(X), \text{ s. t. } q_X R_{\text{conf}}^{[\mathcal{A}]} q_X' \text{ then } q_X R_{\text{strict}}^{[\mathcal{A}]} q_X' \).
- (no exclusive creation by \( \mathcal{A} \)) \( \forall q_X \in \text{states}(X), \forall a \in \text{sig}(X)(q_X) \) \( \mathcal{A}\text{-}exclusive \) in \( q_X \), \( \text{created}(X)(q_X)(a) = \emptyset \).

This definition 6.13 allows the next definition 6.14 to be well-defined. A \( \mathcal{A}\text{-}conservative \) PCA is a \( \mathcal{A}\text{-}fair \) PCA that does not hide any output action that could be an external action of \( \mathcal{A} \). This assumption will allow us to define the composition between \( \mathcal{A} \) and \( Y = X \setminus \{ \mathcal{A} \} \) in the proof of monotonicity.

**Definition 6.14 (\( \mathcal{A}\text{-}conservative \) PCA).** Let \( X \) be a PCA, \( \mathcal{A} \in \autids \). We say that \( X \) is \( \mathcal{A}\text{-}conservative \) if it is \( \mathcal{A}\text{-}fair \) and for every state \( q_X, C_X = \text{config}(X)(q_X) \) s. t. \( \mathcal{A} \in \text{aut}(C_X) \) and \( \text{map}(C_X)(\mathcal{A}) = q_\mathcal{A} \), hidden-actions(X)(q_X) \( \cap \) \( \overline{\text{ext}(\mathcal{A})(q_\mathcal{A})} = \emptyset \).

### 6.3 Corresponding w. r. t. \( \mathcal{A}, \mathcal{B} \)

We are closed to state all the technical assumptions to achieve monotonicity of print-implementation w.r.t. PSIOA creation. We introduce one last assumption so-called creation-explicitness, used in section 11 to reduce implementation of \( X_B \) by \( X_A \) to implementation of \( \mathcal{B} \) by \( \mathcal{A} \).

Intuitively, a PCA is \( \mathcal{A}\text{-}creation-explicit \) if the creation of a sub-automaton \( \mathcal{A} \) is equivalent to the triggering of an action in a dedicated set. This property will allow to obtain the reduction of lemma 11.23.

**Definition 6.15 (creation-explicit PCA).** Let \( \mathcal{A} \) be a PSIOA and \( X \) be a PCA. We say that \( X \) is \( \mathcal{A}\text{-}creation-explicit \) iff: it exists a set of actions, noted creation-actions(\( X \))(\( \mathcal{A} \)), s. t. \( \forall q_X \in \text{states}(X), \forall a \in \text{sig}(X)(q_X), \text{ if we note } A_X = \text{auts}(\text{config}(X)(q_X)) \) and \( \varphi_X = \text{created}(X)(q_X)(a) \), then \( \mathcal{A} \notin A_X \land \mathcal{A} \in \varphi_X \iff a \in \text{creation-actions}(X)(\mathcal{A}) \).

Now we can define new (non naively) correspondence w. r. t. PSIOA \( \mathcal{A}, \mathcal{B} \) to define (non naively) monotonic relationship.

**Definition 6.16 (corresponding w. r. t. \( \mathcal{A}, \mathcal{B} \)).** Let \( \mathcal{A}, \mathcal{B} \in \autids \), \( X_A \) and \( X_B \) be PCA we say that \( X_A \) and \( X_B \) are corresponding w. r. t. \( \mathcal{A}, \mathcal{B} \), if 1) they are naively corresponding w. r. t. \( \mathcal{A}, \mathcal{B} \), 2) they are \( \mathcal{A}\text{-}conservative \) and \( \mathcal{B}\text{-}conservative \) respectively and 3) they are \( \mathcal{A}\text{-}creation explicit \) and \( \mathcal{B}\text{-}creation explicit \) with creation-actions(\( X_A \))(\( \mathcal{A} \)) = creation-actions(\( X_B \))(\( \mathcal{B} \)) respectively i. e. they verify:

- \( X_A \) is \( \mathcal{A}\text{-}conservative \) and \( X_B \) is \( \mathcal{B}\text{-}conservative \)
- \( X_A \) is \( \mathcal{A}\text{-}creation explicit \) and \( X_B \) is \( \mathcal{B}\text{-}creation explicit \) with creation-actions(\( X_A \))(\( \mathcal{A} \)) = creation-actions(\( X_B \))(\( \mathcal{B} \))
- \( \text{config}(X_A)(q_{X_A}) \preceq_{\mathcal{A}\mathcal{B}} \text{config}(X_B)(q_{X_B}) \).
- \( X_A, X_B \) are creation-corresponding w.r.t. \( \mathcal{A}, \mathcal{B} \)
- \( X_A, X_B \) are hiding-corresponding w.r.t. \( \mathcal{A}, \mathcal{B} \)
- (No creation from \( \mathcal{A} \) and \( \mathcal{B} \))
  - \( \forall q_{X_A} \in \text{states}(X_A) \), for every action \( act \) \( \mathcal{A}\text{-}exclusive \), \( \text{created}(X_A)(q_{X_A})(act) = \emptyset \) and similarly
Definition 6.17 (Monotonic relationship). Let $R$ be a binary relation on PSIOA. We say that $R$ is monotonic if for every pair of PSIOA $(\mathcal{A}, \mathcal{B}) \in R$, for every pair of PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ that are corresponding w. r. t. $\mathcal{A}$, $\mathcal{B}$, $(psioa(X_{\mathcal{A}}), psioa(X_{\mathcal{B}})) \in R$.

We would like states the monotonicity of print-implementation, but it holds only for a certain class of schedulers, so-called creation-oblivious introduced in next subsection 6.4

6.4 Creation-oblivious scheduler

Here we present a particular scheduler schema, that do not take into account previous internal actions of a particular sub-automaton to output its probability over transitions to trigger.

We start by defining strict oblivious-schedulers that output the same transition with the same probability for pair of execution fragments that differ only by prefixes in the same class of equivalence. This definition is inspired by the one provided in the thesis of Segala, but is more restrictive since we require a strict equality instead of a correlation (section 5.6.2 in [8]).

Definition 6.18 (oblivious scheduler). Let $\tilde{W}$ be a PCA or a PSIOA, let $\tilde{\sigma} \in \text{schedulers}(\tilde{W})$ and let $\equiv$ be an equivalence relation on $\text{Frags}^*(\tilde{W})$ verifying $\forall \tilde{\alpha}_1, \tilde{\alpha}_2 \in \text{Frags}^*(\tilde{W})$ s. t. $\tilde{\alpha}_1 \equiv \tilde{\alpha}_2$, $\text{Istate}(\tilde{\alpha}_1) = \text{Istate}(\tilde{\alpha}_2)$ . We say that $\tilde{\sigma}$ is ($\equiv$)-strictly oblivious if $\forall \tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3 \in \text{Frags}^*(\tilde{W})$ s. t. 1) $\tilde{\alpha}_1 \equiv \tilde{\alpha}_2$ and 2) $\tilde{fstate}(\tilde{\alpha}_3) = \text{Istate}(\tilde{\alpha}_2) = \text{Istate}(\tilde{\alpha}_1)$, then $\tilde{\sigma}(\tilde{\alpha}_1 \tilde{\alpha}_3) = \tilde{\sigma}(\tilde{\alpha}_2 \tilde{\alpha}_3)$.

Now we define the relation of equivalence that defines our subset of creation-oblivious schedulers. Intuitively, two executions fragments ending on $\mathcal{A}$ creation are in the same equivalence class if they differ only in terms of internal actions of $\mathcal{A}$.

Definition 6.19 ($\equiv_{\mathcal{A}}^{cr}$ $\tilde{\alpha}$ $\tilde{\alpha}'$). Let $\mathcal{A}$ be a PSIOA, and $\tilde{W}$ be a PCA. For every $\tilde{\alpha}, \tilde{\alpha}' \in \text{Frags}^*(\tilde{W})$, we say $\tilde{\alpha} \equiv_{\mathcal{A}}^{cr} \tilde{\alpha}'$ iff:

1. $\tilde{\alpha}, \tilde{\alpha}'$ both ends on $\mathcal{A}$-creation.
2. $\tilde{\alpha}$ and $\tilde{\alpha}'$ differ only in the $\mathcal{A}$-exclusive actions and the states of $\mathcal{A}$, i.e. $\mu(\tilde{\alpha}) = \mu(\tilde{\alpha}')$ where $\mu(\tilde{\alpha} = q_0 q_1 q_2 \ldots q_n) \in \text{Frags}^*(\tilde{W})$ is defined as follows:
   - remove the $\mathcal{A}$-exclusive actions
   - replace each state $\tilde{q}_i$ by its configuration $Config(\tilde{W})(\tilde{q}_i) = (A^i, S^i)$
   - replace each configuration $(A^i, S^i)$ by $(A^i, S^i) \setminus \{\mathcal{A}\}$
   - replace the (non-alternating) sequences of identical configurations (due to $\mathcal{A}$-exclusiveness of removed actions) by one unique configuration.
3. $\text{trace}(\tilde{\alpha}) = \text{trace}(\tilde{\alpha}')$.
4. $\text{Istate}(\tilde{\alpha}) = \text{Istate}(\tilde{\alpha}')$

We can remark that the items 4 can be deduced from 1 and 2 if $X$ is configuration-conflict-free.

Definition 6.20 (creation-oblivious scheduler). Let $\mathcal{A}$ be a PSIOA, $\tilde{W}$ be a PCA, $\tilde{\sigma} \in \text{schedulers}(\tilde{W})$. We say that $\tilde{\sigma}$ is $\mathcal{A}$-creation oblivious if it is ($\equiv_{\mathcal{A}}^{cr}$)-strictly oblivious.

We say that $\tilde{\sigma}$ is creation-oblivious if it is $\mathcal{A}$-creation oblivious for every sub-automaton $\mathcal{A}$ of $\tilde{W}$ ($\mathcal{A} \in \bigcup \{\text{states}(\tilde{W})\}$ $\text{auts}(Config(\tilde{W})(q))$). We note $\text{CrOb}$ the function that maps any PCA $\tilde{W}$ to the set of creation-oblivious schedulers of $\tilde{W}$.  

- $\forall q_{X_B} \in \text{states}(X_B)$, for every action $act'$ $\mathcal{B}$-exclusive, $\text{created}(X_B)(q_{X_B})(act') = \emptyset$
We have formally defined our notion of creation-oblivious scheduler. This will be a key property to ensure lemma 11.23 that allows to reduce the measure of a class of comportment as a function of measures of classes of shorter comportment where no creation of $A$ or $B$ occurs excepting potentially at very last action. This reduction is more or less necessary to obtain monotonicity of implementation relation:

**Theorem 6.21 ($\leq_{CrOb}^{\text{print}}$ is monotonic).** Let $\mathcal{A}, \mathcal{B} \in \text{Autids}$, $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ be PCA corresponding w. r. t. $\mathcal{A}$, $\mathcal{B}$. If $\mathcal{A} \leq_{CrOb}^{\text{print}} \mathcal{B}$, then $X_{\mathcal{A}} \leq_{CrOb}^{\text{print}} X_{\mathcal{B}}$.

The remaining sections are dedicated to the proof of this theorem 6.21. We start by defining in section 7 a morphism between executions of automata, so called executions-matching, that preserves structure and measure of probability under *alter ego schedulers*. Next, we define in section 8 the notion of an automaton $X_{\mathcal{A}}$ derived from a PSIOA $\mathcal{A}$, noted $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$. Furthermore, we show in section 9 that there is an executions-matching from a PCA $X_{\mathcal{A}}$ to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\hat{\mathcal{A}}$ where $\hat{\mathcal{A}}$ is the singleton wrapper of $\mathcal{A}$, i.e. a PCA that only handle $\mathcal{A}$. The section 11 uses the morphism of section 9 to reduce the implementation of $X_{\mathcal{B}}$ by $X_{\mathcal{A}}$ to the implementation of $\mathcal{B}$ by $\mathcal{A}$ and finally obtain the monotonicity of implementation w.r.t. PSIOA creation. Finally section 12 explains why the task-scheduler introduced in [4] is not creation-oblivious.

### 7 EXECUTIONS-MATCHING

In this section, we introduce some tools to formalise the fact that two automata have the same comportment for the same scheduler. This section is composed by two sub-sections on PSIOA executions-matching and PCA executions-matching. Basically, an executions-matching execution from an automaton $\mathcal{A}$ to another automaton $\mathcal{B}$ is a morphism $f^{ex}$ from Execs($\mathcal{A}$) to Execs($\mathcal{B}$) that is structure-preserving. In the remaining, we will often use an executions-matching to show that a pair of executions $(\alpha, \pi = f^{ex}(\alpha)) \in \text{Execs}(\mathcal{A}) \times \text{Execs}(\mathcal{B})$ have the same probability $\epsilon_\pi(\alpha) = \epsilon_\pi'(\pi)$ under a pair of so-called *alter-ego* schedulers $(\sigma, \sigma') \in \text{schedulers}(\mathcal{A}) \times \text{schedulers}(\mathcal{B})$ that have corresponding comportment after corresponding executions fragment $(\alpha', \pi' = f^{ex}(\alpha')) \in \text{Frags}^*(\mathcal{A}) \times \text{Frags}^*(\mathcal{B})$.

#### 7.1 PSIOA executions-matching and semantic equivalence

This first subsection is about PSIOA executions-matching.

*matching execution.* An executions-matching need a states-matching (see definition 7.1) and a transitions-matching (see definition 7.3) to be defined itself.

**Definition 7.1 (states-matching).** Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA with $Q_{\mathcal{A}} = \text{states}(\mathcal{A})$ and $Q_{\mathcal{B}} = \text{states}(\mathcal{B})$ as respective sets of states, let $Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$ and let $f : Q'_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a mapping that verifies:

- Starting state preservation: If $\tilde{q}_{\mathcal{A}} \in Q'_{\mathcal{A}}$ then $f(\tilde{q}_{\mathcal{A}}) = \tilde{q}_{\mathcal{B}}$ (with $(\tilde{q}_{\mathcal{A}}, \tilde{q}_{\mathcal{B}}) = (\text{start}(\mathcal{A}), \text{start}(\mathcal{B}))$)
- Signature preservation (modulo an hiding operation): $\forall (q, q') \in Q'_{\mathcal{A}} \times Q_{\mathcal{B}}$, s. t. $q' = f(q)$, sig($\mathcal{A}$)($q$) = hide(sig($\mathcal{B}$)($q'$), h($q'$)) with h($q'$) $\subseteq$ out($\mathcal{B}$)($q'$) (resp. with h($q'$) = $\emptyset$, that is sig($\mathcal{A}$)($q$) = sig($\mathcal{B}$)($q'$)).

then we say that $f$ is a weak (resp. strong) states-matching from $\mathcal{A}$ to $\mathcal{B}$. If $Q'_{\mathcal{A}} = Q_{\mathcal{A}}$, then we say that $f$ is a complete (weak or strong) states-matching from $\mathcal{A}$ to $\mathcal{B}$.

Before being able to define transitions-matching, some requirements have to be ensured. A set of transition that would ensure these requirements would be called *eligible to transitions-matching.*
Definition 7.2 (transitions set eligible to transitions matching). Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA with $Q_\mathcal{A} = \text{states}(\mathcal{A})$ and $Q_\mathcal{B} = \text{states}(\mathcal{B})$ as respective sets of states, let $Q'_\mathcal{A} \subset Q_\mathcal{A}$ and let $f : Q'_\mathcal{A} \to Q_\mathcal{B}$ be a states-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $D'_\mathcal{A} \subseteq D(\mathcal{A}) = \text{dtrans}(\mathcal{A})$ be a subset of transition. If $D'_\mathcal{A}$ verifies that for every $(q,a,\eta(\mathcal{A},q,a)) \in D'_\mathcal{A}$:

- Matched states preservation: $q \in Q'_\mathcal{A}$ and

- Equitable corresponding distribution: $\forall q'' \in \text{supp}(\eta(\mathcal{A},q,a))$, $q'' \in Q'_\mathcal{A}$ and $\eta(\mathcal{A},q,a) \xrightarrow{f} \eta(\mathcal{B},f(q,a))$ (i.e. the restriction of $f$, $\tilde{f} : \text{supp}(\eta(\mathcal{A},q,a)) \to \text{supp}(\eta(\mathcal{B},f(q,a)))$ is bijective and $\forall q'' \in \text{supp}(\eta(\mathcal{A},q,a))$, $\eta(\mathcal{A},q,a)(q'') = \eta(\mathcal{B},f(q,a))(f(q''))$).

then we say that $D'_\mathcal{A}$ is eligible to transitions-matching domain from $f$. We omit to mention the states-matching $f$ when this is clear in the context.

Now, we are able to define a transitions-matching, which is a property-preserving mapping from a set of transitions $D'_\mathcal{A} \subseteq \text{dtrans}(\mathcal{A})$ to another set of transitions $D'_\mathcal{B} \subseteq \text{dtrans}(\mathcal{B})$.

Definition 7.3 (transitions-matching). Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA with $Q_\mathcal{A} = \text{states}(\mathcal{A})$ and $Q_\mathcal{B} = \text{states}(\mathcal{B})$ as respective sets of states, let $Q'_\mathcal{A} \subset Q_\mathcal{A}$ and let $f : Q'_\mathcal{A} \to Q_\mathcal{B}$ be a states-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $D'_\mathcal{A} \subseteq D(\mathcal{A})$ be a subset of transition eligible to transitions-matching domain from $f$.

We define the transitions-matching $(f,f^{tr})$ from $\mathcal{A}$ to $\mathcal{B}$ induced by the states-matching $f$ and the subset of transition $D'_\mathcal{A}$ s.t. $f^{tr} : D'_\mathcal{A} \to D(\mathcal{B})$ is defined by $f^{tr}((q,a,\eta(\mathcal{A},q,a))) = (f(q),a,\eta(\mathcal{B},f(q,a)))$. If $f$ is complete and $D'_\mathcal{A} = D(\mathcal{A})$, $(f,f^{tr})$ is said to be a complete transitions-matching. If $f$ is weak (resp. strong) $(f,f^{tr})$ is said to be a weak (resp. strong) transitions-matching. If $f$ is clear in the context, with a slight abuse of notation, we say that $f^{tr}$ is a transitions-matching.

The function $f^{tr}$ need to verify some constraints imposed by $f$, but if the set $D'_\mathcal{A}$ of concerned transitions is correctly-chosen to ensure the 2 properties of definition 7.2, then such a transitions-matching is unique.

Now, we can easily define an executions-matching with a transitions-matching, which is a property-preserving mapping from a set of execution fragments $F'_\mathcal{A} \subseteq \text{Frags}(\mathcal{A})$ to another set of execution fragments $F'_\mathcal{B} \subseteq \text{Frags}(\mathcal{B})$.

Definition 7.4 (executions-matching). Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $(f,f^{tr})$ be a transitions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $F'_\mathcal{A} = \{ a_1 \pm q_1^a q_1^1...a^n q^n_1 \in \text{Frags}(\mathcal{A}) | \forall i \in [0 : |a| - 1], (q^i_a,a_i^{a+1},\eta(\mathcal{A},q^i_a,a_i^{a+1})) \in \text{dom}(f^{tr}) \}$. Let $f^{ex} : F'_\mathcal{A} \to \text{Frags}(\mathcal{B})$, built from $(f,f^{tr})$ s.t. $\forall a = q_0^a q_1^a...a^n q^n_a \in F'_\mathcal{A}$, $f^{ex}(a) = f(q^0_0,a)^1 f(q^0_1)...a^n f(q^n_a)$.

We say that $(f,f^{tr},f^{ex})$ is an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Furthermore, if $(f,f^{tr})$ is complete and $F'_\mathcal{A} = \text{Frags}(\mathcal{A})$, $(f,f^{tr},f^{ex})$ is said to be a complete executions-matching. If $(f,f^{tr})$ is weak (resp. strong) $(f,f^{tr},f^{ex})$ is said to be a weak (resp. strong) executions-matching. When $(f,f^{tr})$ is clear in the context, with a slight abuse of notation, we say that $f^{ex}$ is an executions-matching.

The function $f^{ex}$ is completely defined by $(f,f^{tr})$, hence we call $(f,f^{tr},f^{ex})$ the executions-matching induced by the transition matching $(f,f^{tr})$ or the executions-matching induced by the states-matching $f$ and the subset of transitions $\text{dom}(f^{tr})$.

The construction of $f^{ex}$ allows us to see two executions mapped by an executions-mapping as a sequence of pairs of transitions mapped by the attached transitions-matching. This result is formalised in next lemma 7.5.

Lemma 7.5 (executions-matching seen as a sequence of transitions-matchings). Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $(f,f^{tr},f^{ex})$ be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $\alpha = q_0^a q_1^a...a^n q^n_a \in \text{dom}(f^{ex})$ and $\pi = f^{ex}(\alpha) = q_0^a q_1^a...a^n q^n_a = f(q^0_0) a f(q^1_1)...a^n f(q^n_a)$. Then for every $i \in [0 : |a|-1]$, $(q^i_0,a_i^{a+1},\eta(\mathcal{B},q^i_0,a_i^{a+1})) = f^{tr}((q^i_0,a_i^{a+1},\eta(\mathcal{A},q^i_0,a_i^{a+1})))$.
PROOF. First, matched states preservation and action preservation are ensured by construction. By definition, for every $i \in [0 : |\alpha| - 1]$, $(q^i_A, a^{i+1}, \eta_{(A,q^i_A,a^{i+1})}) \in \text{dom}(f^{tr})$. We note $\text{tr}^i_B = f^{tr}((q^i_A, a^{i+1}, \eta_{(A,q^i_A,a^{i+1})}))$. By definition, $\text{tr}^i_B$ is of the form $(f(q^i_A), a^{i+1}, \eta)$. But a transition of this form is unique, which means $\text{tr}^i_B = (f(q^i_A), a^{i+1}, \eta_{(B,f(q^i_A),a^{i+1})})$ which ends the proof. □

Matching executions

Fig. 18. Here we have $Q'_A = \{q^0, q^1, ..., q^9\} \subseteq Q_A$, we define the state-matching $f : Q'_A \rightarrow Q_B$ s. t. $\forall k \in [1, 9], f(q^k) = \tilde{q}^k$, and $D'_A = \{(q^0, a, \eta_{(A,q^0,a)}), (q^1, b, \eta_{(A,q^1,b)}), (q^2, c, \eta_{(A,q^2,c)}), (q^3, d, \eta_{(A,q^3,d)}), (q^4, e, \eta_{(A,q^4,e)}), (q^5, f, \eta_{(A,q^5,f)}), (q^6, g, \eta_{(A,q^6,g)}), (q^7, h, \eta_{(A,q^7,h)}), (q^8, i, \eta_{(A,q^8,i)})\}$. We can define the execution matching $(f, f^{tr}, f^{ex})$ induced by $f$ and $D'_A$.

Now we overload the definition of executions-matching to be able to state the main result of this paragraph i.e. theorem 7.9

Definition 7.6 (executions-matching overload: pre-execution-distribution). Let $A$ and $B$ be two PSIOA. Let $(f, f^{tr}, f^{ex})$ be an executions-matching from $A$ to $B$.

Let $(\mu, \mu') \in \text{Disc}(\text{Frags}(A)) \times \text{Disc}(\text{Frags}(B))$ s. t. $\mu \mapsto f^{ex}$ (i.e. the restriction of $f^{ex}$ on supp($\mu$) is a bijection from supp($\mu$) to supp($\mu'$) and for every $\alpha \in \text{supp}(\mu)$, $\mu'(f^{ex}(\alpha)) = \mu(\alpha)$). Then we say that $(f, f^{tr}, f^{ex})$ is an executions-matching from $(A, \mu)$ to $(B, \mu')$.  

41
We define the \( t \cdot \) (the sense that their outputs depends only on the semantic structure of the input, preserved by the executions-matching. executions-matching preserves measure of probability of the corresponding executions.

**Definition 7.7 (Continued executions-matching).** Let \( A \) and \( B \) be two PSIOA with \( Q_A = \text{states}(A) \) and \( Q_B = \text{states}(B) \) as respective sets of states and \( D_A = \text{dtrans}(A) \) and \( D_B = \text{dtrans}(B) \) as respective sets of discrete transitions. Let \( (f, f^{tr}, f^{ex}) \) be an executions-matching from \( A \) to \( B \) with \( \text{dom}(f) \supseteq Q_A \) and \( \text{dom}(f^{tr}) \supseteq D_A \).

Let \( f^+ : Q_A'' \rightarrow Q_B \) with \( Q_A'' \subseteq Q_A \). Let \( D_A' \subseteq \text{dtrans}(A) \) be a subset of transitions verifying for every \( (q,a,\eta_{(A,q,a)}) \in D_A' \setminus D_A'\) :

- Matched states preservation: \( q \in Q_A' \)
- Extension of equitable corresponding distribution: \( \forall q'' \in \text{supp}(\eta_{(A,q,a)}), q'' \in Q_A'' \) and \( \eta_{(A,q,a)} \xrightarrow{f^+} \eta_{(B,f(q),a)} \), i.e. the restriction of \( f^+ \) on \( \text{supp}(\eta_{(A,q,a)}) \), \( f^+ : \text{supp}(\eta_{(A,q,a)}) \rightarrow \text{supp}(\eta_{(B,f(q),a)}) \) is bijective and \( \forall q'' \in \text{supp}(\eta_{(A,q,a)}), \eta_{(A,q,a)}(q'') = \eta_{(B,f(q),a)}(f^+''(q'')) \)

We define the \((f^+, D_A''\)-continuation of \( f^{tr} \) as the function \( f^{tr,+} : D_A' \cup D_A'' \rightarrow D_B \) s.t. \( \forall (q,a,\eta_{(A,q,a)}) \in D_A' \cup D_A'', f^{tr,+}(\eta_{(A,q,a)}) = (f(q),a,\eta_{(B,f(q),a)}).

Let \( f''_A = \text{dom}(f^{ex}) \cup \{ \alpha \in \text{Execx}(A) | \alpha \in \text{dom}(f^{ex}) \wedge (q,a,\eta_{(A,q,a)}) \in D_A'' \} \). We define the \((f^{tr,+}, f^{ex,+})\)-continuation of \( f^{ex} \) as the function \( f^{ex,+} : f''_A \rightarrow \text{Frags}(B) \) s.t. \( \forall \alpha \in \text{dom}(f^{ex}), f^{ex,+}(\alpha) = f^{ex}(\alpha) \) and \( \forall \alpha' = \alpha^{-} q,a,q' \in f''_A \setminus \text{dom}(f^{ex}), f^{ex,+}(\alpha') = f^{ex}(\alpha')^{f(q),a,f^{+}(q')}.\)

Then, we say that \(((f,f^{tr}), f^{tr,+}, f^{ex,+})\) is the \((f^+, D_A'')\)-continuation of \((f, f^{tr}, f^{ex})\) which is a continuation of \((f, f^{tr}, f^{ex})\) and a continued executions-matching from \( A \) to \( B \).

Moreover, if \((\mu, \mu') \in \text{Disc}(\text{Frags}(A)) \times \text{Disc}(\text{Frags}(B)) \) s.t. \( \mu \xrightarrow{f^{ex,+}} \mu' \) (i.e. the restriction \( f^{ex,+} \) of \( f^{ex} \) on \( \text{supp}(\mu) \) is a bijection from \( \text{supp}(\mu) \) to \( \text{supp}(\mu') \) and for every \( \alpha \in \text{supp}(\mu) \), \( \mu'(f^{ex,+}(\alpha)) = \mu(\alpha) \)), then we say that \(((f,f^{tr}), f^{tr,+}, f^{ex,+})\) is a continued executions-matching from \( (A, \mu) \) to \( (B, \mu') \).

**From executions-matching to probabilistic distribution preservation.** We want to states that a (potentially-continued) executions-matching preserves measure of probability of the corresponding executions.

To do so, we define alter egos schedulers to a certain executions-matching. Such pair of schedulers are very similar in the sense that their outputs depends only on the semantic structure of the input, preserved by the executions-matching.

**Definition 7.8 ((f, f^{tr}, f^{ex})-alter egos schedulers).** Let \( A \) and \( B \) be two PSIOA. Let \((f, f^{tr}, f^{ex})\) be an executions-matching from \( A \) to \( B \). Let \((\delta, \sigma) \in \text{schedulers}(A) \times \text{schedulers}(B) \). We say that \((\delta, \sigma) \) are \((f, f^{tr}, f^{ex})\)-alter egos (or \( f^{ex}\)-alter egos) if, and only if, for every \((\alpha, \sigma) \in \text{Frags}(A) \times \text{Frags}(B) \) s.t. \( \alpha = f^{ex}(\alpha) \) (which means \( \text{sig}(A)(q) = \text{sig}(B)(q) \) \( \equiv \text{sig}(\delta) \) with \( \delta = \text{Istate}(\delta) \) and \( q = \text{Istate}(\alpha) \) by signature preservation property of the associated states-matching), \( \forall a \in \text{sig}(\delta)((\delta, a, \eta_{(A,q,a)})) = \sigma(a)((q, a, \eta_{(B,q,a)})). \)

Let us remark that the previous definition implies that the probability of halting after corresponding executions fragments \((\delta, \alpha) \) is also the same.

Now we are ready to states an intuitive result that will be often used in the remaining.

**Theorem 7.9 (Executions-matching preserves general probabilistic distribution).** Let \( A \) and \( B \) be two PSIOA. Let \((\mu, \mu') \in \text{Disc}(\text{Frags}(A)) \times \text{Disc}(\text{Frags}(B)). \) Let \((f, f^{tr}, f^{ex})\) be an executions-matching from \( (A, \mu) \) to \( (B, \mu') \). Let \((\delta, \sigma) \) are \((f, f^{tr}, f^{ex})\)-alter egos. Let \((\delta, \alpha) \in \text{Frags}(A) \times \text{Frags}(B) \) s.t. \( \alpha = f^{ex}(\alpha) \). Then \( \epsilon_{\delta,\mu}(C_{\alpha}) = \epsilon_{\sigma,\mu'}(C_{\alpha}) \) and \( \epsilon_{\delta,\mu'}(\delta) = \epsilon_{\sigma,\mu}(\delta) \).
We want to show that \( \varepsilon \) \( f' \) \( \varepsilon \) \( f \) \( \varepsilon \) Dynamic Probabilistic Input Output Automata be two PSIOA. Let \((\tilde{a}, a_0) \in supp(\mu)\) with \(\forall \tilde{a}_o \in supp(\mu), \mu(f^x(\tilde{a}_o)) = \mu(\tilde{a}_o)\). Second, by definition 6.13 of measure generated by a scheduler, \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \sum_{a_o \in supp(\mu)} \mu(a_o) \cdot \varepsilon_{\sigma,\mu}(C_{\alpha'})\) and \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \sum_{a_o \in supp(\mu)} \mu(a_o) \cdot \varepsilon_{\sigma,\mu}(C_{\alpha'})\). Hence, by combining (\(\cdot\)) and (\(\cdot\)), we only need to show that for every \((\tilde{a}_o, a_0) \in supp(\mu) \times supp(\mu)\) with \(f^x(\tilde{a}_o) = a_o\), for every \((\tilde{a}', a') \in Frags^*(A) \times Frags^*(B)\) with \(f^x(\tilde{a}') = a'\), we have \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\) that we show by induction on the size \(s = |\tilde{a}| = |a|\). We fix \((\tilde{a}, a_0) \in supp(\mu) \times supp(\mu)\) with \(f^x(\tilde{a}_o) = a_o\).

**Basis:** \(s = 0\)

Let \(\tilde{a}' = \tilde{q}' \in Frags^*(A), a' = q' \in Frags^*(B)\) with \(a' = f^x(\tilde{a}')\). We have \(|\tilde{a}'| = |a'| = 0\). By definition 6.13 of measure generated by a scheduler,

\[
\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \begin{cases} 
0 & \text{if both } \alpha' \neq a_o \text{ and } a_o \neq \tilde{a}' \\
1 & \text{if } \tilde{a}' \leq a_o \\
\varepsilon_{\sigma,\mu}(C_{\alpha'}) \cdot \sigma(\tilde{a})(\eta(\tilde{a}, a_0)) \cdot \eta(\tilde{a}, a_0)(q') & \text{if } a_o \leq \tilde{a}' \text{ and } \alpha' \neq \tilde{a}' \end{cases}
\]

Since \(|\tilde{a}'| = |a'| = 0\), the third case is never met. The second case can be written: \(\tilde{a}' \leq a_o\) iff \(f_{\text{state}}(a_o) = \tilde{q}'\) (resp. \(f_{\text{state}}(a_o) = q'\)). Hence, for every \((\tilde{a}_o, a_0)\) s.t. \(f^x(\tilde{a}_o) = a_o, \varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\) which ends the basis.

**Induction:** We assume the result to be true up to size \(s\) and we show it implies the result is true for size \(s + 1\). Let \((\tilde{a}', \tilde{a}', a') \in Frags^*(A) \times Frags^*(B)\) with \(\tilde{a}' = \tilde{a}' \tilde{q}a'q'\) and \(a' = a''\tilde{q}a'q'\). By definition 6.13 of measure generated by a scheduler,

\[
\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \begin{cases} 
0 & \text{if both } \alpha' \neq a_o \text{ and } a_o \neq \tilde{a}' \\
1 & \text{if } \alpha' \leq a_o \\
\varepsilon_{\sigma,\mu}(C_{\alpha'}) \cdot \sigma(\tilde{a})(\eta(\tilde{a}, a_0)) \cdot \eta(\tilde{a}, a_0)(q') & \text{if } a_o \leq \tilde{a}' \text{ and } \alpha' \neq \tilde{a}' \end{cases}
\]

Again, the executions-matching implies that i) both \(\tilde{a}' \neq a_o\) and \(a_o \neq \tilde{a}'\) both \(\alpha' \neq a_o\) and \(a_o \neq a'\), ii) \(\tilde{a} \leq a_o\) \(\iff\) \(a \leq a_o\) and \(i\) \(\tilde{a} \leq a_o\) \(\iff\) \(a \leq a_o\) \(\iff a \leq a\). Moreover, by induction assumption \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\). Hence we only need to show that \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\). By definition of alter-ego schedulers, \(\sigma(\tilde{a})(\eta(\tilde{a}, a_0)) \cdot \eta(\tilde{a}, a_0)(q') = \sigma(\alpha)(\eta(a, a_0)) \cdot \eta(\tilde{a}, a_0)(q')\). By definition of executions-matching, \(\eta(\tilde{a}, a_0)(q') = \eta(\tilde{a}, a_0)(q')\). Thus (i) and (ii) implies which allows us to conclude that \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\).

Finally, let \(\text{sig} = \text{sig}(A)(\text{Istate}(a')) = \text{sig}(A)(\text{Istate}(a'))\), then \(\varepsilon_{\sigma,\mu}(\tilde{a}') = \varepsilon_{\sigma,\mu}(C_{\alpha'}) \cdot (1 - \sum_{a \in \text{sig}} g(a') \cdot a) = \varepsilon_{\sigma,\mu}(C_{\alpha'}) \cdot (1 - \sum_{a \in \text{sig}} g(a') \cdot a) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\), which ends the proof.

We restate the previous theorem with continued executions-matching.

**Theorem 7.10** *(Contd executions-matching preserves general probabilistic distribution).* Let \(A\) and \(B\) be two PSIOA. Let \((\tilde{a}, a_0) \in \text{Disc}(\text{Frags}(A)) \times \text{Disc}(\text{Frags}(B))\). Let \((f, f^r, f^x)\) be an executions-matching from \((A, \tilde{a})\) to \((B, \mu)\). Let \((f, f^r, f^x)\) be a continuation of \((f, f^r, f^x)\). Let \((\tilde{a}, \sigma) \in \text{Schedulers}(A) \times \text{Schedulers}(B), s.t. \(\tilde{a}, \sigma)\) are \((f, f^r, f^x)\)-alter egos. Let \((\tilde{a}, a') \in Frags^*(A) \times Frags^*(B)\). s.t. \(a = f^x(\tilde{a})\). Then \(\varepsilon_{\sigma,\mu}(C_{\alpha'}) = \varepsilon_{\sigma,\mu}(C_{\alpha'})\).
Proof. The proof is exactly the same than the one for theorem 7.9 \hfill \square

Before dealing with composability of executions-matching, we prove two results about injectivity and surjectivity of executions-matching in next lemma 7.11 and 7.12.

Lemma 7.11 (Injectivity of executions-matching). Let \((f, f^r, f^\text{ex})\) be an executions-matching from \(A\) to \(B\) and \(((f, f^r), f^{r+}, f^{\text{ex}+})\) a continuation of \((f, f^r, f^\text{ex})\).

Let \(\tilde{f}^\text{ex+} : F''_A \subseteq \text{dom}(f^{\text{ex}+}) \rightarrow \tilde{F}_B \subseteq \text{range}(f^{\text{ex}+})\). Let \(\tilde{f} : Q''_A \subseteq \text{dom}(f) \rightarrow \text{states}(B)\) be the restriction of \(f\) on a set \(Q''_A \subseteq \text{dom}(f)\).

(1) If \(i) \forall \alpha \in F''_A\), \(f\text{state}(\alpha) \in Q''_A\) and \(ii) \tilde{f}\) is injective, then \(\tilde{f}^{\text{ex}+}\) is injective.

(2) (Corollary) If \(F''_A \subseteq \text{Execs}(A)\), \(f^{\text{ex}+}\) is injective.

Proof. (1) By induction on the size \(k\) of the prefix: Basis: By i) \(f\text{state}(a)\), \(f\text{state}(a') \in Q''_A\), by construction of \(f^{\text{ex}+}\), \(f(f\text{state}(a)) = f(f\text{state}(a')) = f(\pi)\) and by ii) \(f\text{state}(a) = f\text{state}(a')\) Induction. We assume the injectivity of \(f^{\text{ex}+}\) to be true for execution on size \(k\) and we show this is also true for size \(k + 1\). Let \(\pi = s^0b^1s^2...s^k\) \((k+1)\) \(\in F''_B\). Let \(\alpha = aq^0a_1q^1...a_kq^{k+1}\) and \(\alpha' = aq^0a_1q^1...q_k a^{k+1}q^{k+1}\) \(\in F''_A\) s.t. \(f(\alpha) = f(\alpha') = \pi\). By construction of \(f^{\text{ex}+}\), \(\forall i \in [1, n], b^i = a^i\). By construction of \(f^{\text{ex}+}\), \(f^{\text{ex}+}(q^0a_1q^1...q_k) = f^{\text{ex}+}(q^0a_1q^1...q_k) = s^0a^1s^2...s^k\). By induction assumption \(q^0a_1q^1...q_k) = q^0a_1q^1...q_k\). By definition of execution, \(s^k \in \text{supp}(\eta(\gamma, a, k+1))\). By equitable corresponding distribution, \(\eta(A, q^{k+1}) \in \text{dom}(f^{tr})\), the restriction of \(f\), \(\tilde{f} : \text{supp}(\eta(A, q^{k+1})) \rightarrow \text{supp}(\eta(B, a, k+1))\) is injective and \(\eta(A, q^{k+1}) \in \text{dom}(f^{tr})\) \(\setminus \text{dom}(f^{tr})\), the restriction of \(f^{tr}\), \(f^{tr} : \text{supp}(\eta(A, q^{k+1})) \rightarrow \text{supp}(\eta(B, a, k+1))\) is injective so \(q^{k+1} = q^{k+1}\) which ends the proof.

(2) We have \(\text{start}(A) = 1\). Hence the restriction of \(f\) on \(\text{start}(A)\) is necessarily injective (ii). Let \(a \in \text{Execs}(A)\).

By definition of execution, \(f\text{state}(a) \in \text{start}(A)\) (i). All the requirements of lemma 7.11, first item are met, which ends the proof. \hfill \square

Lemma 7.12 (Surjectivity property preserved by continuation). Let \(A\) and \(B\) be two PSIOA. Let \((f, f^r, f^\text{ex})\) be an executions-matching from \(A\) to \(B\). Let \(((f, f^r), f^{r+}, f^{\text{ex}+})\) be the counterpart of \((f, f^r, f^\text{ex})\) (where by definition \(D''_A \setminus \text{dom}(f^r)\) respect the properties of matched states preservation and extension of equitable corresponding distribution from definition 7.7). If the restriction \(f^\text{ex} : E'_A \subseteq \text{Execs}(A) \rightarrow E_B \subseteq \text{Execs}(B)\) is surjective, then \(f^{\text{ex}+} : E''_A \rightarrow \tilde{E}_B\) is surjective.

Proof. Let \(\pi' = \pi^{-} q_{B} \in \text{Execs}(B)\) s.t. \(\pi' \in E_B\) and \(\exists \alpha \in \text{supp}(\eta(A, q_{A}, a, q_{B})) \ni \text{Istate}(a)\) (and \(q_{A}, a, \eta(A, q_{A}, a) \in \text{dom}(f^{tr+})\)). By definition of \(f^{tr+}\), \(f^{tr+}(\eta(A, q_{A}, a, q_{B})) \rightarrow \eta_{A, q_{A}, a} \in \text{dom}(f^{tr})\) \(\setminus \text{dom}(f^{tr})\) \(\eta_{A, q_{A}, a} \xrightarrow{f^{tr}} \eta_{B, q_{B}, a}\) and if ii) \(\eta(A, q_{A}, a, q_{B}) \in \text{dom}(f^{tr+})\) \(\eta_{A, q_{A}, a} \xrightarrow{f^{tr+}} \eta_{B, q_{B}, a}\) \(\eta_{A, q_{A}, a} \xrightarrow{f^{tr}} \eta_{B, q_{B}, a}\). In both cases, it exists \(q'_{A} \in \text{supp}(\eta(A, q_{A}, a, q_{B}))\) s.t. \(f^{\text{ex}+}(\pi' = f^{\text{ex}+}(\pi' = \pi^{-} q_{B} \in \text{Execs}(B))\) s.t. \(\pi' \in E_B\) and \(\exists \alpha \in \text{supp}(\eta(A, q_{A}, a, q_{B})) \ni \text{Istate}(a)\) (and \(q_{A}, a, \eta(A, q_{A}, a) \in \text{dom}(f^{tr+})\)).

We finish this paragraph with the concept of semantic equivalence that describes a pair of PSIOA that differ only syntactically.

Definition 7.13 (semantic equivalence). Let \(A\) and \(B\) be two PSIOA. We say that \(A\) and \(B\) are semantically-equivalent if it exists \(f : \text{Execs}(A) \rightarrow \text{Execs}(B)\) which is a complete bijective executions-matching from \(A\) to \(B\).
**Composability of execution-matching relationship.** Now we are looking for composability of executions-matching. First we define natural extension of notions presented in previous paragraph for the automaton obtained after composition with another automaton $E$.

**Definition 7.14 ($E$-extension).** Let $A$ and $B$ be two PSIOA with $Q_A = states(A)$ and $Q_B = states(B)$ as respective sets of states and let $E$ be partially-compatible with both $A$ and $B$.

1. Let $Q_A' \subset Q_A$. We call $E$-extension of $Q_A'$ the set of states $Q_A'||E = \{ q \in states(A||E) | q \uparrow A \in Q_A' \}$
2. Let $f : Q_A' \subset Q_A \rightarrow Q_B$. We call $E$-extension of $f$ the function $g : Q_A'||E \rightarrow states(B) \times states(E)$ s. t. $\forall (q_A, q_E) \in Q_A'||E, g((q_A, q_E)) = (f(q_A), q_E)$
3. Let $D_A' \subset D_A$ a subset of transitions. We call $E$-extension of $D_A'$ the set $D_A'||E = \{ ((q_A, q_E), a, \eta((A, E),(q_A, q_E), a)) \in dtrans(A||E) | q_A \in Q_A'$ and either $(q_A, a, \eta((A, E),(q_A, q_E), a)) \in D_A'$ or the action $a$ is not enabled in $q_A$.

Now, we can start with the composability of states-matching.

**Lemma 7.15 (Composability of states-matching).** Let $A$ and $B$ be two PSIOA with $Q_A = states(A)$ and $Q_B = states(B)$ as respective sets of states. Let $E$ be partially-compatible with $A$ and $B$. Let $f : Q_A' \subset Q_A \rightarrow Q_B$ be a states-matching. Let $g$ be the $E$-extension of $f$.

If $range(g) \subset states(B||E)$, then $g$ is a states-matching from $A||E$ to $B||E$.

**Proof.**

- Starting state preservation: if $(\bar{q}_A, \bar{q}_E) \in Q_A||E$ then $\bar{q}_A \in Q_A'$ which means $f(\bar{q}_A) = \bar{q}_B$, thus $g((\bar{q}_A, \bar{q}_E)) = (\bar{q}_B, \bar{q}_E)$.
- Signature preservation (modulo an hiding operation): $\forall ((q_A, q_E), (q_B, q_E)) \in Q_A'||E \times states(B||E)$ with $(q_B, q_E) = g((q_A, q_E))$, we have $\text{sig}(A)(q_A) = \text{sig}(B)(f(q_A)) = \text{hide}(\text{sig}(B)(q_B), h(q_B))$ with $h(q_B) \subseteq \text{out}(B)(q_B)$.

Since $A$ and $E$ are partially-compatible, $\text{sig}(A)(q_A) = \text{hide}(\text{sig}(B)(q_B), h(q_B))$ is compatible with $\text{sig}(E)(q_E)$ which means a fortiori $\text{sig}(B)(q_B)$ is compatible with $\text{sig}(E)(q_E)$.

Namely $\forall act \in h(q_B), act \notin \text{in}(E)(q_E)$. Hence $\text{sig}(A)(E)(q_A, q_E)) = \text{hide}(\text{sig}(B, E)(q_B, q_E), h'(q_B, q_E))$ with $h'(q_B, q_E)) = h(q_B) \subseteq \text{out}(B)(q_B) \subseteq \text{out}(E)(q_B, q_E))$ which ends the proof.

The composability of states-matching is ensured under the condition $range(g) \subset states(B||E)$ where $g$ is the $E$-extension of the original states-matching $f : Q_A' \subseteq states(A) \rightarrow states(B)$. In next lemma, we give a sufficient condition to ensure $range(g) \subset states(B||E)$. This is the one that we will use in practice.

**Definition 7.16 (reachable-by and states of execution (recall)).** Let $A$ be a PSIOA or a PCA. Let $E_A' \subseteq \text{Execs}(A)$. We note reachable-by $E_A = \{ q \in states(A) | \exists \alpha \in E_A', lstate(\alpha) = q \}$. Let $\alpha = q^0, a^1, q^2, \ldots, a^n, q^n$. We note states($\alpha$) = $\cup_{i \in [0..n]} q^i$.

**Lemma 7.17 (A sufficient condition to obtain range(g) \subset states(B||E)).** Let $A$ and $B$ be two PSIOA with $Q_A = states(A)$ and $Q_B = states(B)$ as respective sets of states. Let $E$ be partially-compatible with both $A$ and $B$. Let $f : Q_A' \subset Q_A \rightarrow Q_B$ be a states-matching. Let $Q_A'||E = E$-extension of $Q_A'$.

Let $Q_A'||E \subset Q_A'||E$, the set of states reachable by an execution that counts only states in $Q_A'||E$. i. e.

- $E_A'||E = \{ \alpha \in \text{Execs}(A||E) | \text{states}(\alpha) \subseteq Q_A'||E \}$
- $Q_A'||E = \text{reachable-by}(E_A'||E)$
Let \( f'' \) the restriction of \( f \) to set \( Q''_A = \{ q_A = ( (q_A, q_E) \mapsto A ) | (q_A, q_E) \in Q'' || E \} \).

Then the \( E \)-extension of \( f'' \), noted \( g'' \), verifies \( \text{range}(g'') \subset \text{states}(B||E) \).

Proof. By induction on the minimum size of an execution \( \tilde{a} = q^0 a^1 \ldots q^n \) with \( q^n = q_0, \forall i \in [0, n], q_i \in Q''_A || E \). Basis \((|\tilde{a}| = 0 \implies a = \tilde{a} \triangleright q_0\)) we consider \( q^n = \tilde{q}_A \). We have \( g((\tilde{\eta}_A, \tilde{q}_E)) = f((\tilde{\eta}_A, \tilde{q}_E)) = (\tilde{q}_B, \tilde{q}_E) \in \text{states}(B||E) \).

We assume this is true for \( \tilde{a} \) with \( \text{Istate}(\tilde{a}) = q \) and we show this is also true for \( \tilde{a}' = \tilde{a} \triangleright qq' \). By induction hypothesis \( q \in \text{states}(B||E) \). Since \( q' \in \text{states}(A||E), A \) and \( E \) are compatible at state \( (q''_A, q''_E) \), that is \( \text{sig}(A)(q''_A) \) and \( \text{sig}(E)(q''_E) \) are compatible, which means that a fortiori, \( (\text{sig}(B)(f''(q''_A))) \) and \( \text{sig}(E)(q''_E) \) are compatible and so \( B \) and \( E \) are compatible at state \( (f''(q''_A), q''_E) = g''(q') \). Hence \( g''(q') \) is a reachable compatible state of \( (B, E) \) which means this is a state of \( B||E \).

Now, we can continue with the composability of transitions-matching.

**Lemma 7.18 (Composability of Eligibility for Transitions-Matching).** Let \( A \) and \( B \) be two PSIOA with \( Q_A = \text{states}(A) \) and \( Q_B = \text{states}(B) \) as respective sets of states. Let \( E \) be partially-compatible with \( A \) and \( B \). Let \( f : Q''_A \subset Q_B \rightarrow Q_B \) be a states-matching and \( D''_A \subset A \) a subset of transitions eligible to transitions-matching domain from \( f \). Let \( g \) be the \( E \)-extension of \( f \) and \( D''_A||E \) the \( E \)-extension of \( D_A \).

If \( \text{range}(g) \subset \text{states}(B||E) \), then \( D''_A||E \) is compatible to transitions-matching domain from \( g \).

Proof. Let \( ((q_A, q_E), a, q_E, \eta((A||E),(q_A, q_E), a)) \in D''_A||E \).

By definition, \( q_A \in Q''_A \) which means \( (q_A, q_E) \in Q''_A||E \), so the matched states preservation is ensured. We still need to ensure the equitable corresponding distribution.

- Let \( (q''_A, q''_E) \in \text{supp}(\eta((A||E),(q_A, q_E), a)) \). If \( a \in \text{sig}(A) \), then \( q''_A \in \text{supp}(\eta((A||E),(q_A, q_E), a)) \) which means \( q''_A \in Q''_A \) and hence \( (q''_A, q''_E) \in Q''_A||E \). If \( a \notin \text{sig}(A), \eta((A||E),(q_A, q_E), a) = \delta q_A, \) which means \( q''_A = q_A \in Q''_A \) and hence \( (q''_A, q''_E) \in Q''_A||E \). Thus for every \( (q''_A, q''_E) \in \text{supp}(\eta((A||E),(q_A, q_E), a)) \), \( (q''_A, q''_E) \in Q''_A||E \).

- \( \eta((A||E),(q_A, q_E), a) = \eta((A||E),(q_A, q_E), a) \) \( (q''_A, q''_E) = \eta((A||E),(q_A, q_E), a) \) \( (q''_A, q''_E) = \eta((A||E),(q_A, q_E), a) \) \( (q''_A, q''_E) = \eta((B||E),(q_A, q_E), a) \) \( (q''_A, q''_E) = \eta((B||E),(q_A, q_E), a) \) \( (q''_A, q''_E) \) \( (q''_A, q''_E) \) \( (q''_A, q''_E) \), which ends the proof of equitable corresponding distribution.

**Definition 7.19 (E-extension of an execution-matching).** Let \( A \) and \( B \) be two PSIOA with \( Q_A = \text{states}(A) \) and \( Q_B = \text{states}(B) \) as respective sets of states. Let \( E \) be partially-compatible with both \( A \) and \( B \). Let \( (f, f^\triangleright, f^\frown) \) be an executions-matching from \( A \) to \( B \). Let \( g \) the \( E \)-extension of \( f \). If \( \text{range}(g) \subset \text{states}(B||E) \), then

1. we call the \( E \)-extension of \( f^\triangleright \) the function \( g^ \triangleright : (q, a, \eta((A||E),(q_A, q_E), a)) \in D''_A||E \mapsto (q, a, \eta((B||E),(g(q), a)))) \) where \( D''_A||E \) is the \( E \)-extension of the domain \( \text{dom}(f^\triangleright) \) of \( f^\triangleright \).
2. we call the \( E \)-extension of \( (f, f^\triangleright, f^\frown) \) the matching-execution \( (g, g^\triangleright, g^\frown) \) from \( A||E \) to \( B||E \) induced by \( g \) and \( \text{dom}(g^\triangleright) \).

Finally we can states the main result of this paragraph, i.e. theorem 7.20 of executions-matching composability.

**Theorem 7.20 (Composability of Executions-Matching).** Let \( A \) and \( B \) be two PSIOA. Let \( E \) be partially-compatible with both \( A \) and \( B \). Let \( (f, f^\triangleright, f^\frown) \) be an execution-matching from \( A \) to \( B \) where \( g \) represents the \( E \)-extension of \( f \). If \( \text{range}(g) \subset \text{states}(B||E) \), then the \( E \)-extension of \( (f, f^\triangleright, f^\frown) \) is a matching-execution \( (g, g^\triangleright, g^\frown) \) from \( A||E \) to \( B||E \) induced by \( g \) and \( \text{dom}(g^\triangleright) \).
We repeated the previous definition, while an executions-matching only need a states-matching $g$ and a set $\text{dom}(g^*)$ of transitions eligible to transitions-matching domain from $g$ which is provided by construction.

Here we give some properties preserved by $E$-extension of an executions-matching.

**Lemma 7.21 (Some properties preserved by $E$-extension of an executions-matching).** Let $A$ (resp. $B$) be a PSSOA with $Q_A$ (resp. $Q_B$) as set of states. Let $(f, f^*, f^x)$ be an execution-matching from $A$ to $B$.

1. If $f$ is bijective and $f^{-1}$ is complete, then for every PSSOA $E$ partially-compatible with $A$, $E$ is partially-compatible with $B$.

2. Let $E$ partially-compatible with both $A$ and $B$, let $g$ be the $E$-extension of $f$.
   - (a) If $f$ is bijective and $f^{-1}$ is complete, then $\text{range}(g) = \text{states}(B|E)$ and so we can talk about the $E$-extension of $(f, f^*, f^x)$.
   - (b) If $(f, f^*, f^x)$ is a bijective complete transition-matching, $(g, g^*, g^x)$ is a bijective complete transition-matching. (And $(f, f^*, f^x)$ and $(g, g^*, g^x)$ are bijective execution-matching.)
   - (c) If $f$ is strong, then $g$ is strong.

3. Let $E$ partially-compatible with both $A$ and $B$, let $g$ be the $E$-extension of $f$. Let assume $\text{range}(g) \subseteq \text{states}(B|E)$. Let $(g, g^*, g^x)$ be the $E$-extension of $(f, f^*, f^x)$.
   - (a) If the restriction $f^x : E' \subseteq \text{Execs}(A) \rightarrow \tilde{E}_B \subseteq \text{Execs}(B)$ is surjective, then $\tilde{f}^x : \{ \alpha \in \text{Execs}(A||E) | \alpha \upharpoonright A \in E' \} \rightarrow \{ \pi \in \text{Execs}(B||E) | \pi \upharpoonright B \in \tilde{E}_B \}$ is surjective.
   - (b) If $f$ is strong, $g$ is strong.

**Proof.**

(1) We need to show that every pseudo-execution of $(B, E)$ ends on a compatible state. Let $\pi = q_0^0 a_1^1 q_1^1 \ldots a_n^1 q_n$ be a finite pseudo-execution of $(B, E)$. We note $\alpha = (f^{-1}(q_0^0), q_0^0, a_1^1 f^{-1}(q_1^1), q_1^1) \ldots a_n^1 f^{-1}(q_n^1), q_n^1)$.

The proof is in two steps. First, we show by induction that $\alpha = (f^{-1}(q_0^0), q_0^0, a_1^1 f^{-1}(q_1^1), q_1^1) \ldots a_n^1 f^{-1}(q_n^1), q_n^1)$ is an execution of $A||E$. Second, we deduce that it means $(f^{-1}(q_0^0), q_0^0)$ is a compatible state of $(A, E)$ which means that a fortiori, $(q_0^0, q_0^1)$ is a compatible state of $(B, E)$ which ends the proof.

- First, we show by induction that $\alpha$ is an execution of $A||E$. We have $(f^{-1}(q_0^0), q_0^0) = (q_0^0, q_0^1)$ which ends the basis.

Let assume $(f^{-1}(q_0^0), q_0^0), a_1^1 f^{-1}(q_1^1), q_1^1) \ldots a_k^1 f^{-1}(q_k^1), q_k^1)$ is an execution of $A||E$. Hence $(f^{-1}(q_0^0), q_0^0), a_1^1 f^{-1}(q_1^1), q_1^1) \ldots a_k^1 f^{-1}(q_k^1), q_k^1)$ is a compatible state of $(A, E)$ which means that a fortiori, $a_k^1 f^{-1}(q_k^1), q_k^1)$ is a compatible state of $(B, E)$ because of signature preservation of $f$.

For the same reason, $\text{sig}(B||E)(q_k^1) = \text{sig}(A, E)((f^{-1}(q_k^1), q_k^1))$, so $a_k^1 \delta f^{-1}(q_k^1), q_k^1))$. Then we use the completeness of $(f^{-1}, (f^*)^{-1})$, to obtain the fact that either $\delta f^{-1}(q_k^1) \in \text{dom}(f^*)^{-1})$ or $a_k^1 \notin \text{sig}(B)(q_k^1)$ (and we recall the convention that in this second case $\eta((B, q_k^1, a_{k+1}^1) = \delta q_k^1)$. which means either $(f^{-1}(q_k^1), a_{k+1}^1, \eta((A, f^{-1}(q_k^1), a_{k+1}^1))$ is a transition of $A$ that ensures any $q'' \in \text{supp}(\eta((B, q_k^1, a_{k+1}^1)))$. Thus for every $(q''', q'''') \in \text{supp}(\eta((A, f^{-1}(q_k^1), a_{k+1}^1)))$, $f^{-1}(q''') = \eta((A, f^{-1}(q_k^1), a_{k+1}^1)$. Hence, $(f^{-1}(q_k^1), a_{k+1}^1)$ is reachable by $(A, E)$ which means $(f^{-1}(q_k^1), a_{k+1}^1) f^{-1}(q_k^1), q_k^1) \ldots a_k^1 f^{-1}(q_k^1), q_k^1)$ is an execution of $A||E$. Thus by induction $a$ is an execution of $A||E$. 

47
(2) (a) Let \((q^0_B, q^0_E) \in \text{states}(B||E)\). This state is reachable, so we note \(n = (q^0_B, q^1_B, q^1_E, \ldots, q^n_B, q^n_E)\) the execution of \(B||E\). Thereafter, we note \(n = (f^{-1}(q^0_B), q^1_B, q^1_E, \ldots, q^n_B, q^n_E)\).

We can show by induction that \(n\) is an execution of \(A||E\). The proof is exactly the same as in 1.

Hence \(n\) is an execution of \(A||E\) which means \((f^{-1}(q^0_B), q^0_E)\) is a state of \(A||E\) and \(g((f^{-1}(q^0_B), q^0_E)) = (q^0_B, q^0_E)\) to finally prove that it exists \(q^*\) s. t. \(g(q^*) = (q^0_B, q^0_E)\) which means \(\text{states}(B||E) \subseteq \text{dom}(g)\).

We can reuse the proof of 1. to show that if \(q \in \text{states}(A||E)\), then \(g(q) \in \text{states}(B||E)\) which means \(\text{dom}(g) \subseteq \text{states}(B||E)\).

Hence dom \(g = \text{states}(B||E)\).

We can apply the previous lemma 7.18 to obtain the eligibility of \(D_{A||E}\).

(b) Let assume \((f, f^{tr})\) be bijective. The bijectivity of \(g\) is immediate \(g(\ldots) = (f(\ldots), Id(\ldots))\). The bijectivity of \(g^{tr}\) is also immediate since \(g^{tr} : \eta(\eta_{A,q,A,a}) \otimes \eta(\eta_{E,q,A,a}) \rightarrow f^{tr} \eta(\eta_{A,q,A,a}) \otimes \eta(\eta_{E,q,A,a})\) with \(f^{tr}\) bijective.

(c) Immediate, since in this case \(\text{sig}(A||E)(q_{A,n}) = \text{sig}(B||E)(q_{A,n})\) implies \(\text{sig}(A||E)(q_{A,n}) = \text{sig}(B||E)(f(q_{A,n}, q_{E,n}))\).

(3) (a) Let \(\pi = ((q^0_B, q^0_E), a^1, (q^1_B, q^1_E), \ldots, a^n, (q^n_B, q^n_E)) \in \text{Exe}(B||E)\) with \(\pi \uparrow B = (q^0_B, a^1, q^1_B, \ldots, a^n, q^n_B) \in E_B\), where the monotonic function \(k : [0, n] \rightarrow [0, m]\), verifies \(\forall i \in [0, n], k(i) \in [0, m], q^i_B = q^k_B\).

By surjectivity of \(f^{tr}\) we have \(\hat{a} = (\hat{q}^0_A, a^1, \hat{q}^1_A, \ldots, a^n, \hat{q}^n_A) \in E_A\) s. t. \(f^{tr}(\hat{a}) = q^0_B \uparrow B\). We note \(a = (q^0_B, q^0_E, a^1, q^1_A, q^1_E, \ldots, a^n, q^n_A, q^n_E)\) where \(\forall i \in [0, n], q^i_A = q^k^i_A\). Hence, \(\forall i \in [0, n], g(q^i_A, q^i_E) = (q^i_B, q^i_E)\).

Moreover, \(\forall i \in [0, n-1], q^{i+1}_A \in \text{supp}(\eta_{(A,q_A,a^i)} \otimes \eta_{(B,q_B,a^i)}\) since \(q^{i+1}_A, q^i_E) \in \text{supp}(\eta_{(B,q_B,a^i)}\).

Finally, by signature preservation, \(\forall i \in [0, n-1], a^{i+1} \in \text{sig}(A)(q^i_A) \cup \text{sig}(E)(q^i_E)\).

We are ready to states the composability of semantic equivalence.

**Theorem 7.22 (Composability of Semantic Equivalence).** Let \(A\) and \(B\) be PSIOA semantically-equivalent. Then for every PSIOA \(E\):

- \(E\) is partially-compatible with \(A \iff E\) is partially-compatible with \(B\).
- \(E\) is partially-compatible with both \(A\) and \(B\), then \(A||E\) and \(B||E\) are semantically-equivalent PSIOA.

**Proof.**

- The first item (\(E\) is partially-compatible with \(A \iff E\) is partially-compatible with \(B\)) comes from lemma 7.21, first item.
- The second item (if \(E\) is partially-compatible with both \(A\) and \(B\), then \(A||E\) and \(B||E\) are semantically-equivalent PSIOA) comes from lemma 7.21, second item.
A weak complete bijective transition-matching implies a weak complete bijective execution-matching which means the two automata are completely semantically equivalent modulo some hiding operation that implies that some PSIOA are partially-compatible with one of the automaton and not with the other and that the traces are not necessarily the same ones.

**Definition 7.23 (E-extension of continued executions-matching).** Let $A$ and $B$ be two PSIOA. Let $E$ be partially-compatible with both $A$ and $B$. Let $(f, f^{tr}, f^{ex})$ be the $(f^{tr}, D^{tr}_{A'})$-continuation of $(f, f^{tr}, f^{ex})$ (where by definition $D^{tr}_{A'} \setminus \text{dom}(f^{tr})$ respect the properties of matched states preservation and extension of equivalent corresponding distribution from definition 7.7). If the respective $E$-extension of $f$ and $f^+$, noted $g$ and $g^+$, verify range($g$) $\cup$ range($g^+$) $\subseteq (B[E)$, we define the $E$-extension of $(f, f^+, f^{tr+}, f^{ex+})$ as $(g, g^+, g^{tr+}, g^{ex+})$, where

- $(g, g^+, g^{tr+}, g^{ex+})$ is the $E$-extension of $(f, f^{tr}, f^+)$
- $g^{tr+} : \{(q, \eta (A||E), q, a) \in D^{tr+}_{A||E} \rightarrow (g(q), a, \eta (A||E), q, a)\}$ where $D^{tr+}_{A||E}$ is the $E$-extension of $\text{dom}(f^{tr+})$
- $\forall a' = \alpha \neg q, q', a$, with $a' \in \text{dom}(g^{ex+})$, if $(q, a, \eta (A||E), q, a) \in \text{dom}(g^{tr+}) \land g^{ex+}(a') = g^{ex}(a)$ and if $(q, a, \eta (A||E), q, a) \in \text{dom}(g^{tr+}) \land g^{ex+}(a') = g^{ex}(a) \land g(q), a, q$

**Lemma 7.24 (Commutativity of continuation and extension).** Let $A$ and $B$ be two PSIOA. Let $E$ be partially-compatible with both $A$ and $B$. Let $(f, f^{tr}, f^{ex})$ be an executions-matching from $A$ to $B$. Let $(f, f^+, f^{tr+}, f^{ex+})$ be the $(f^+, D^{tr+}_{A'})$-continuation of $(f, f^{tr}, f^{ex})$ (where by definition $D^{tr+}_{A'}$ respect the properties of matched states preservation and extension of equivalent corresponding distribution from definition 7.7). Let

- $(g, g^+, g^{tr+}, g^{ex+})$ be the $E$-extension of $(f, f^{tr}, f^+)$ verifying range($g$) $\subseteq \text{states}(B[E)$, 
- $D^{tr+}_{A||E}$ the $E$-extension of $\text{dom}(f^{tr+})$, i.e. $D^{tr+}_{A||E} = \{(q, q', a, \eta (A||E,(q, a, q'), a)) \in \text{dtrans}(A||E) \land \text{dom}(f) \land \text{dtrans}(A||E) \land \text{dom}(f)\}$
- $g^{tr+}_{(A||E)}$ the $E$-extension of $f^+$

Then

1. $D^{tr+}_{A||E} \setminus \text{dom}(g^{tr+})$ verifies matched states preservation and extension of equivalent corresponding distribution.
2. The $(g^{tr+}_{(A||E)}(D^{tr+}_{A||E}))$-continuation of $(g, g^{tr+}, g^{ex+})$, noted $(g, g^{tr+}, g^{ex+})$, is equal to the $E$-extension of $(f, f^{tr}, f^{tr+}, f^{ex+})$, noted $(g, g^{tr+}, g^{ex+})$.

We show that the operation of continuation and extension are in fact commutative.

**Proof.** We start by showing $D^{tr+}_{A||E} \setminus \text{dom}(g^{tr+})$ verifies matched states preservation and extension of equivalent corresponding distribution. By definition 7.7 of $E$-extension, $D^{tr+}_{A||E} = \{(q, q', a, \eta (A||E,(q, a, q'), a)) \in \text{dtrans}(A||E) \land \text{dom}(f) \land \text{dtrans}(A||E) \land \text{dom}(f)\}$, while $g^{tr+} = \{(q, q', a, \eta (A||E,(q, a, q'), a)) \in \text{dtrans}(A||E) \land \text{dom}(f) \land \text{dtrans}(A||E) \land \text{dom}(f)\}$. Thus $D^{tr+}_{A||E} \setminus \text{dom}(g^{tr+}) = \{(q, q', a, \eta (A||E,(q, a, q'), a)) \in \text{dtrans}(A||E) \land \text{dom}(f) \land \text{dtrans}(A||E) \land \text{dom}(f)\}$. Let $\text{tr} = \{(q, q', a, \eta (A||E,(q, a, q'), a)) \in D^{tr+}_{A||E} \setminus \text{dom}(g^{tr+})$, then

- Matched states preservation: By (*) $q_A \in \text{dom}(f)$ which leads immediately to $(q, q', a, \eta (A||E,(q, a, q'), a) \in \text{dtrans}(A||E) \land \text{dom}(f)$
According to the definition of \(\Sigma\), the states are ensured by the definition of complete bijective execution-matching after renaming. Let \(A\) and \(B\) be two PSIOAs such that \(\Sigma_A\) and \(\Sigma_B\) are the states of \(A\) and \(B\) respectively, with \(\text{ren}(A)\) and \(\text{ren}(B)\) being the renaming functions. We have shown that \(D_{\Sigma_A}^{\text{fin}}(\Sigma_B)\), the set of final states, is preserved under renaming.

Now, we show another property.

- By definition 7.7 of continuation, \(g_{\Sigma(c,e)}^+ = g_{\Sigma(c,e)}^+\).

  By definition 7.7 of continuation, \(\text{dom}(g_{\Sigma(c,e)}^+) = \text{dom}(g^+) \cup D_{\Sigma_A}^{\text{fin}}(\Sigma_B)\) and \(D_{\Sigma_B}^{\text{fin}}(\Sigma_A)\) are disjoint.

- By definition 7.19, \(\text{dom}(g_{\Sigma(c,e)}^+) = \text{dom}(g_{\Sigma(c,e)}^+) \cup D_{\Sigma_A}^{\text{fin}}(\Sigma_B)\) and \(D_{\Sigma_B}^{\text{fin}}(\Sigma_A)\) are disjoint.

- We prove that \(g_{\Sigma(c,e)}^+ = g_{\Sigma(c,e)}^+\), where \(\Sigma_A\) and \(\Sigma_B\) are disjoint.

- By definition 7.19, \(\text{dom}(g_{\Sigma(c,e)}^+) = \text{dom}(g_{\Sigma(c,e)}^+) \cup D_{\Sigma_A}^{\text{fin}}(\Sigma_B)\) and \(D_{\Sigma_B}^{\text{fin}}(\Sigma_A)\) are disjoint.

- By definition 7.7 of continuation, \(\text{dom}(g_{\Sigma(c,e)}^+) = \text{dom}(g_{\Sigma(c,e)}^+) \cup D_{\Sigma_A}^{\text{fin}}(\Sigma_B)\) and \(D_{\Sigma_B}^{\text{fin}}(\Sigma_A)\) are disjoint.

- We can remark that properties of equitable corresponding distribution are not conflicting since \(\text{dom}(g_{\Sigma(c,e)}^+) = \text{dom}(g^+)\).

- \(g_{\Sigma(c,e)}^+\) and \(g_{\Sigma(c,e)}^+\) are entirely defined by \((g, g_{\Sigma(c,e)}^+), (g^+, g_{\Sigma(c,e)}^+)\) and \((g, g_{\Sigma(c,e)}^+), (g^+, g_{\Sigma(c,e)}^+)\) that are equal.

**Application for renaming and hiding.** Before dealing with PCA-executions-matching, we state two intuitive theorems of executions-matching after renaming and hiding operations.

**Theorem 7.25.** (strong complete bijective execution-matching after renaming) Let \(A\) and \(B\) be two PSIOAs and \(\text{ren} : \text{states}(A) \rightarrow \text{states}(B)\) s.t. \(\text{ren}(A)\). The ren\(\text{ren}^\text{fin}\), ren\(\text{ren}^\text{ex}\) is a strong complete bijective execution-matching from \(A\) to \(B\) with \(\text{dom}(\text{ren}\text{ren}^\text{fin}) = D_A \equiv \text{dtrans}(A)\).

**Proof.** By definition ren ensures starting state preservation and strong signature preservation. By definition ren is a complete bijection, which implies matched state preservation. The equitable corresponding distribution is also ensured by definition of ren. Hence, all the properties are ensured.

**Theorem 7.26.** (weak complete bijective execution-matching after hiding) Let \(A\) be a PSIOA. Let \(h\) defined on states\(\text{(A)}\), s.t. \(\forall q \in \text{states}(A), q(h) \subseteq \text{out}(A)(q)\). Let \(\text{B} = \text{hiding}(A, h)\). Let \(Id\) the identity function from states\(\text{(A)}\) to states\(\text{(B)}\). Then \(Id, Id^\text{fin}, Id^\text{ex}\) is a weak complete bijective execution-matching from \(A\) to \(B\).

**Proof.** By definition \(Id\) ensures starting state preservation and weak signature preservation. By definition \(Id\) is a complete bijection, which implies matched state preservation. The equitable corresponding distribution is also ensured by definition of hiding. Hence, all the properties are ensured.
7.2 PCA-matching execution

Here we extend the notion of executions-matching to PCA. In practice, we will build executions-matchings that preserve the sequence of configurations visited by concerned executions. Hence, the definition of PCA states-matching is slightly more restrictive to capture this notion of configuration equivalence (modulo action hiding operation), while the other definitions are exactly the same ones.

Definition 7.27 (PCA states-matching). Let \( X \) and \( Y \) be two PCA with \( Q_X = \text{states}(X) \) and \( Q_Y = \text{states}(Y) \) as respective sets of states, \((q_X, q_Y) = (\text{start}(X), \text{start}(Y))\) and let \( f : Q'_X \subset Q_X \rightarrow Q_Y \) be a mapping s. t.:

- Starting state preservation: If \( q_X \in Q'_X \), then \( f(q_X) = q_Y \).
- Configuration preservation (modulo hiding): \( (q_X, q_Y) \in Q'_X \times Q_Y, \) s. t. \( q' = f(q) \), if \( \text{auto}(\text{config}(X)(q)) = (\mathcal{A}_1, ..., \mathcal{A}_n) \), then \( \text{auto}(\text{config}(Y)(q')) = (\mathcal{A}'_1, ..., \mathcal{A}'_n) \) where \( \forall i \in [1 : n], \mathcal{A}_i = \text{hide}(\mathcal{A}'_i, h_i) \) with \( h_i \) defined on \( \text{states}(\mathcal{A}'_i) \), s. t. \( h_i(q_{\mathcal{A}'_i}) \subseteq \text{out}(\mathcal{A}'_i)(q_{\mathcal{A}'_i}) \) (resp. s. t. \( h_i(q_{\mathcal{A}'_i}) = \emptyset \), that is \( \mathcal{A}_i = \mathcal{A}'_i \))
- Hiding preservation (modulo hiding): \( (q_X, q_Y) \in Q'_X \times Q_Y, \) s. t. \( q' = f(q) \), \( \text{hidden-actions}(X)(q) = \text{hidden-actions}(Y)(q') \cup h^+(q') \) where \( h^+ \) defined on \( \text{states}(Y) \), s. t. \( h^+(q_Y) \subseteq \text{out}(Y)(q_Y) \) (resp. s. t. \( h^+(q_Y) = \emptyset \), that is \( \text{hidden-actions}(X)(q) = \text{hidden-actions}(Y)(q') \))
- Creation preservation \( (q_X, q_Y) \in Q'_X \times Q_Y, \) s. t. \( q' = f(q) \), \( \forall a \in \text{sig}(X)(q) = \text{sig}(Y)(q') \), \( \text{created}(X)(q)(a) = \text{created}(Y)(q')(a) \).

then we say that \( f \) is a weak (resp. strong) PCA states-matching from \( X \) to \( Y \). If \( Q'_X = Q_X \), then we say that \( f \) is a complete (weak or strong) PCA states-matching from \( X \) to \( Y \).

We naturally obtain that a PCA states-matching is a PSIOA states-matching:

Lemma 7.28 (A PCA states-matching is a PSIOA states-matching). If \( f \) is a weak (resp. strong) PCA states-matching from \( X \) to \( Y \), then \( f \) is a PSIOA states-matching from psioa\((X)\) to psioa\((Y)\) (in the sense of definition 7.1). (The converse is not necessarily true.)

Proof. The signature preservation immediately comes from the configuration preservation and the hiding preservation. \( \square \)

Now, all the definitions from definition 7.2 to definition 7.4 of previous subsections are the same that is:

Definition 7.29 (PCA transitions-matching and PCA executions-matching). Let \( X \) and \( Y \) be two PCA with \( Q_X = \text{states}(X) \) and \( Q_Y = \text{states}(Y) \) as respective sets of states and let \( f : Q'_X \subset Q_X \rightarrow Q_Y \) be a PCA states-matching from \( X \) to \( Y \).

- Let \( D'_X \subseteq D_X = \text{dtrans}(X) \) be a subset of transitions, \( D'_X \) is eligible to PCA transitions-matching domain from \( f \) if it is eligible to PSIOA transitions-matching domain from \( f \) according to definition 7.2.
- Let \( D'_X \subseteq D_X = \text{dtrans}(X) \) be a subset of transitions eligible to PCA transitions-matching domain from \( f \). We define the PCA transitions-matching \((f, f'')\) induced by the PCA states-matching \( f \) and the subset of transitions \( D'_X \) as the PSIOA transitions-matching induced by the PSIOA states-matching \( f \) and the subset of transitions \( D'_X \) according to definition 7.3.
Let $f^{tr} : D_X \subseteq D_X = d\text{trans}(X) \to D_Y$ s. t. $(f, f^{tr})$ is a PCA transitions-matching, we define the PCA executions-matching $(f, f^{tr}, f^{ex})$ induced by $(f, f^{tr})$ (resp. by $f$ and $\text{dom}(f^{tr})$) as the PSIOA executions-matching $(f, f^{tr}, f^{ex})$ induced by $(f, f^{tr})$ (resp. by $f$ and $\text{dom}(f^{tr})$) according to definition 7.4. Furthermore, let $(\mu, \mu') \in \text{Disc}(\text{Frag}(X)) \times \text{Disc}(\text{Frag}(Y))$ s. t. for every $\alpha' \in \text{supp}(\mu), \alpha' \in \text{dom}(f^{ex})$ and $\mu(\alpha) = \mu'(f^{ex}(\alpha'))$.

Then we say that $(f, f^{tr}, f^{ex})$ is a PCA executions-matching from $(X, \mu)$ to $(Y, \mu')$ according to definition 7.6.

The $(f^+, D_X^\mu)$-continuation of a PCA-executions-matching $(f, f^{tr}, f^{ex})$ is the $(f^+, D_X^\mu)$-continuation of $(f, f^{tr}, f^{ex})$ in the according to definition 7.7.

We restate the theorem 7.9 and 7.10 for PCA executions-matching:

**Theorem 7.30** (PCA-execution-matching preserves probabilistic distribution). Let $X$ and $Y$ be two PCA $(\mu, \mu') \in \text{Disc}(\text{Frag}(X)) \times \text{Disc}(\text{Frag}(Y))$. Let $(f, f^{tr}, f^{ex})$ be a PCA executions-matching from $(X, \mu)$ to $(Y, \mu')$. Let $(\tilde{\sigma}, \sigma) \in \text{scheduler}(\mathcal{A}) \times \text{scheduler}(\mathcal{B})$, s. t. $(\tilde{\sigma}, \sigma)$ are $(f, f^{tr}, f^{ex})$-alter egos. Let $(\alpha, \pi) \in \text{dom}(f^{ex}) \times \text{Frag}(Y)$.

If $\pi = f^{ex}(\alpha)$, then $\epsilon_{\tilde{\sigma}, \sigma}(C_{\alpha}) = \epsilon_{\sigma, \pi}(C_{\alpha})$ and $\epsilon_{\tilde{\sigma}, \sigma}(\tilde{\alpha}) = \epsilon_{\sigma, \pi}(\alpha)$.

**Proof.** We just re-apply the theorem 7.9, since $(f, f^{tr}, f^{ex})$ is a PSIOA executions-matching from $(\text{psio}(X), \mu)$ to $(\text{psio}(Y), \mu')$. $\square$

**Theorem 7.31** (Continued PCA executions-matching preserves general probabilistic distribution). Let $X$ and $Y$ be two PCA $(\mu, \mu') \in \text{Disc}(\text{Frag}(X)) \times \text{Disc}(\text{Frag}(Y))$. Let $(f, f^{tr}, f^{ex})$ be a PCA executions-matching from $(X, \mu)$ to $(Y, \mu')$. Let $((f, f^{tr}), f^{ex,+})$ be a continuation of $(f, f^{tr}, f^{ex})$. Let $(\tilde{\sigma}, \sigma) \in \text{scheduler}(\mathcal{A}) \times \text{scheduler}(\mathcal{B})$, s. t. $(\tilde{\sigma}, \sigma)$ are $(f, f^{tr}, f^{ex,+})$-alter egos. Let $(\alpha, \pi) \in \text{dom}(f^{ex,+}) \times \text{Frag}(Y)$.

If $\pi = f^{ex,+}(\alpha)$, then $\epsilon_{\tilde{\sigma}, \sigma}(C_{\alpha}) = \epsilon_{\sigma, \pi}(C_{\alpha})$.

**Proof.** We just re-apply the theorem 7.10 since $((f, f^{tr}), f^{ex,+})$ is a continued PSIOA executions-matching from $(\text{psio}(X), \mu)$ to $(\text{psio}(Y), \mu')$. $\square$

**Composability of execution-matching relationship.** Now we are looking for composability of PCA executions-matching. Here again the notions are the same as the ones for PSIOA excepting for states-matching and for partial-compatibility. Hence we only need to show that i) the $E$-extension of a PCA states-matching is still a PCA states-matching (see lemma 7.32), ii) if $f : \text{states}(X) \to \text{states}(Y)$ is a bijective PCA states-matching and $f^{-1}$ is complete, then for every PCA $E$ partial-compatible with $X, E$ is partial-compatible $Y$ (see lemma 7.34).

**Lemma 7.32** (Composability of PCA states-matching). Let $X$ and $Y$ be two PCA with $Q_X = \text{states}(X)$ and $Q_Y = \text{states}(Y)$ as respective sets of states. Let $E$ be partially-compatible with both $X$ and $Y$. Let $f : Q_X \subseteq Q_X \to Q_Y$ be a PCA states-matching. Let $g$ be the $E$-extension of $f$.

If $\text{range}(g) \subseteq \text{states}(Y||E)$, then $g$ is a PCA states-matching from $X||E$ to $Y||E$.

**Proof.**

- If $(\hat{q}_X, \hat{q}_E) \in Q_X \times Q_E$ then $\hat{q}_X \in Q'_X$ which means $f(\hat{q}_X) = \hat{q}_Y$, thus $g((\hat{q}_X, \hat{q}_E)) = (\hat{q}_E, \hat{q}_E)$.

- $\forall((q_X, q_E), (q_Y, q_E)) \in Q_X \times \text{states}(Y||E)$ with $(q_Y, q_E) = g((q_X, q_E))$, we have
  - Configuration preservation (modulo hiding): if $\text{auto}(\text{config}(X)(q_X)) = (A_1, ..., A_n)$, then $\text{auto}(\text{config}(Y)(q_Y)) = (A_1', ..., A_n')$ where $\forall i \in [1 : n]$, $A_i = \text{hide}(A_i', h_i)$ with $h_i$ defined on states($A_i'$), s. t. $h_i(q_{A_i'}) \subseteq \text{out}(A_i')(q_{A_i'})$ (resp. s. t. $h_i(q_{A_i'}) = 0$, that is $A_i = A_i'$). Hence if $\text{auto}(\text{config}(X||E)(q_{X, q_E})) = (A_1, ..., A_n, B_1, ..., B_m)$, then $\text{auto}(\text{config}(Y||E)(q_{Y, q_E})) = (A_1', ..., A_n', B_1, ..., B_m)$ where $\forall i \in [1 : n]$, $A_i = \text{hide}(A_i', h_i)$ with $h_i$ defined on states($A_i'$), s. t. $h_i(q_{A_i'}) \subseteq \text{out}(A_i')(q_{A_i'})$ (resp. s. t. $h_i(q_{A_i'}) = 0$, that is $A_i = A_i'$).
- Hiding preservation (modulo hiding): \(\text{hidden-actions}(X)(q_X) = \text{hidden-actions}(Y)(q_Y) \cup h^+(q_Y)\) where \(h^+\) defined on \(\text{states}(Y)\), s. t. \(h^+(q_Y) \subseteq \text{out}(q_Y)\). Hence \(\text{hidden-actions}(X||E)((q_X, q_E)) = \text{hidden-actions}(X)(q_X) \cup \text{hidden-actions}(E)(q_E) = \text{hidden-actions}(Y)(q_Y) \cup \text{hidden-actions}(E)(q_E) \cup h^+(q_Y) = \text{hidden-actions}(Y||E)((q_Y, q_E))\) where \(h^+\) defined on \(\text{states}(Y||E)\), s. t. \(h^+(q_Y, q_E)\) = \(h^+(q_Y) \subseteq \text{out}(q_Y)\) \(\subseteq \text{out}(Y||E)((q_Y, q_E))\).

- Creation preservation \(\forall a \in \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y), \text{created}(X)(q_X)(a) = \text{created}(Y)(q_Y)(a)\). Hence \(\forall a \in \text{sig}(X||E)((q_X, q_E)) = \text{sig}(Y||E)((q_Y, q_E))\), either
  * if \(a \in \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y)\) but \(a \notin \text{sig}(E)(q_E)\) and then \(\text{created}(X||E)((q_X, q_E))(a) = \text{created}(X)(q_X)(a) = \text{created}(Y)(q_Y)(a) = \text{created}(Y||E)((q_Y, q_E))(a)\)
  * or if \(a \notin \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y)\) but \(a \in \text{sig}(E)(q_E)\) and then \(\text{created}(X||E)((q_X, q_E))(a) = \text{created}(E)(q_E)(a) = \text{created}(Y||E)((q_Y, q_E))(a)\)
  * or if \(a \in \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y)\) and \(a \in \text{sig}(E)(q_E)\) and then \(\text{created}(X||E)((q_X, q_E))(a) = \text{created}(X)(q_X)(a) \cup \text{created}(E)(q_E)(a) = \text{created}(Y)(q_Y)(a) \cup \text{created}(E)(q_E)(a) = \text{created}(Y||E)((q_Y, q_E))(a)\)

Thus, \(\forall a \in \text{sig}(X||E)((q_X, q_E)) = \text{sig}(Y||E)((q_Y, q_E))\), \(\text{created}(X)(q_X)(a) = \text{created}(Y)(q_Y)(a)\).

\[\square\]

We restate the theorem 7.20 of executions-matching composability.

**Theorem 7.33 (Composability of PCA matching-execution).** Let \(X \text{ and } Y\) be two PCA. Let \(E\) be partially-compatible with both \(X\) and \(Y\). Let \((f, f^{tr}, f^{ex})\) be a PCA executions-matching from \(X\) to \(Y\). Let \(g\) be the \(E\)-extension of \(f\). If \(\text{range}(g) \subseteq \text{states}(Y||E)\), then the \(E\)-extension of \((f, f^{tr}, f^{ex})\) is a PCA executions-matching \((g, g^{tr}, g^{ex})\) from \(X||E\) to \(Y||E\) induced by \(g\) and \(\text{dom}(g^{tr})\).

**Proof.** This comes immediately from theorem 7.20.

\[\square\]

We extend the lemma 7.21 but we have to take a little precaution for the partial-compatibility since here the configurations have to be pairwise compatible, not only the signatures.

**Lemma 7.34 (Some properties preserved by \(E\)-extension of a PCA executions-matching).** Let \(X \text{ and } Y\) be two PCA. Let \((f, f^{tr}, f^{ex})\) be a PCA executions-matching from \(X\) to \(Y\).

1. If \(f\) is complete, then for every PSIOA \(E\) partially-compatible with \(X, E\) is partially-compatible with \(Y\).
2. Let \(E\) partially-compatible with both \(X\) and \(Y\), let \(g\) be the \(E\)-extension of \(f\).
   a. If \(f\) is bijective and \(f^{-1}\) is complete, then \(\text{range}(g) = \text{states}(Y||E)\) and so we can talk about the \(E\)-extension of \((f, f^{tr}, f^{ex})\)
   b. If \((f, f^{tr})\) is a bijective complete transition-matching, \((g, g^{tr})\) is a bijective complete transition-matching.
      (And \((f, f^{tr}, f^{ex})\) and \((g, g^{tr}, f^{ex})\) are bijective complete execution-matching. )
   c. If \(f\) is strong, then \(g\) is strong

**Proof.** (1) We need to show that every pseudo-execution of \((Y, E)\) ends on a compatible state. Let \(\pi = q^0a_1q^1\ldots a^nq^n\) be a finite pseudo-execution of \((Y, E)\). We note \(\alpha = (f^{-1}(q^0_Y), q^0_E)a_1(f^{-1}(q^1_Y), q^1_E)\ldots a^n(f^{-1}(q^n_Y), q^n_E)\). The proof is in two steps. First, we show by induction that \(\alpha = (f^{-1}(q^0_Y), q^0_E)a_1(f^{-1}(q^1_Y), q^1_E)\ldots a^n(f^{-1}(q^n_Y), q^n_E)\) is an execution of \(X||E\). Second, we deduce that it means \((f^{-1}(q^0_Y), q^0_E)\) is a compatible state of \((X, E)\) which means that a fortiori, \((q^0_Y, q^0_E)\) is a compatible state of \((Y, E)\) which ends the proof.

53
First, we show by induction that \( \alpha \) is an execution of \( X \| E \). We have \((f^{-1}(q_Y), q_E) = (\bar{q}_X, \bar{q}_E)\) which ends the basis.

Let assume \((f^{-1}(q_Y^0), q_E^0))a^i(f^{-1}(q_Y^i), q_E^i)...a^k(f^{-1}(q_Y^k), q_E^k)\) is an execution of \( X \| E \). Hence \((f^{-1}(q_Y^k), q_E^k)\) is a compatible state of \((X, E)\) which means a fortiori \( q_Y^k \) is a compatible state of \((Y, E)\) because of signature preservation of \( f \).

For the same reason, \( \overline{\text{sig}}(Y, E)(q_Y^k) = \overline{\text{sig}}(X \| E)((f^{-1}(q_Y^k), q_E^k)) \), so \( a^{k+1} \in \overline{\text{sig}}(X, E)((f^{-1}(q_Y^k), q_E^k)) \). Then we use the completeness of \((f^{-1}, (ftr)^{-1})\), to obtain the fact that either \( \eta(Y, q_Y^k, a^{k+1}) \in \text{dom}((ftr)^{-1}) \) or \( a^{k+1} \not\in \overline{\text{sig}}(Y) \). (and we recall the convention that in this second case \( \eta(Y, q_Y^k, a^{k+1}) = \delta_Y(q_Y^k) \).)

Thus for every \((q''', q''') \in \text{supp}(\eta(Y, q_Y^k, a^{k+1})) \), \((f^{-1}(q'''), q''') = g^{-1}((q'', q''')) \in \text{supp}(\eta(X, \overline{\text{sig}}(Y), a^{k+1})) \) namely for \((q'', q''') = (q_Y^{k+1}, q_E^{k+1}) \). Hence, \((f^{-1}(q_Y^{k+1}), q_E^{k+1})\) is reachable by \((X, E)\) which means \((f^{-1}(q_Y^0), q_E^0))a^0(f^{-1}(q_Y^0), q_E^0)...a^k(f^{-1}(q_Y^k), q_E^k)a^k\) is an execution of \((X, E)\). Thus by induction \( \alpha \) is an execution of \( X \| E \).

Since \( X \) and \( E \) are partially-compatible \((f^{-1}(q_Y^0), q_E^0)\) is a state of \( X \| E \), so \((f^{-1}(q_Y^0), q_E^0)\) is a compatible state of \((X, E)\) which means \((q_Y^0, q_E^0)\) is a fortiori a compatible state of \((Y, E)\). Hence every reachable state of \((Y, E)\) is compatible which means \( Y \) and \( E \) are partially compatible which ends the proof.

(2) This comes immediately from lemma 7.21 since \((f, ftr, f^{tr})\) is a PSIOA executions-matching from \( psioa(X) \) to \( psioa(Y) \) by construction.

\[ \square \]

Finally, we restate the semantic-equivalence.

A strong complete bijective transitions-matching implies a strong complete bijective executions-matching which means the two automata are completely semantically equivalent.

**Definition 7.35 (PCA semantic equivalence).** Let \( X \) and \( Y \) be two PCA. We say that \( X \) and \( Y \) are semantically-equivalent if it exists a complete bijective strong PCA executions-matching from \( X \) to \( Y \) for every \( PSIOA E \):

- \( E \) is partially-compatible with \( X \) \iff \( E \) is partially-compatible with \( Y \)
- if \( E \) is an environment for both \( X \) and \( Y \), then \( X || E \) and \( Y || E \) are PCA semantically-equivalent.

**Theorem 7.36 (Composability of Semantic Equivalence).** Let \( X \) and \( Y \) be PCA semantically-equivalent. Then for every \( PSIOA E \):

- \( E \) is partially-compatible with \( X \) \iff \( E \) is partially-compatible with \( Y \)
- if \( E \) is an environment for both \( X \) and \( Y \), then \( X || E \) and \( Y || E \) are PCA semantically-equivalent.

**Proof.**

- The first item comes from lemma 7.34, first item
- The second item comes from lemma 7.34, second item

\[ \square \]

A weak complete bijective PCA transitions-matching implies a weak complete bijective PCA executions-matching which means the two automata are completely semantically equivalent modulo some hiding operation that implies that some PSIOA are partially-compatible with one of the automaton and not with the other one and that the traces are not necessarily the same ones.
8 PROJECTION

This section aims to formalise the idea of a PCA $X_{\mathcal{A}}$ considered without an internal PSIOA $\mathcal{A}$. This PCA will be noted $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$. The reader can already take a look on the figures 26 and 27 to get an intuition on the desired result. This is an important step in our reasoning since we will be able to formalise in which sense $X_{\mathcal{A}}$ and $\text{psioa}(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\mathcal{A}$ are similar.

We first define some notions of projection on configurations on subsection 8.1. Then we define the notion of $\mathcal{A}$-fair PCA $X$ in subsection 8.2, which will be a sufficient condition to ensure that $Y = X \setminus \{\mathcal{A}\}$ is still a PCA, namely that it ensures the constraints of top/down and bottom/up transition preservation, which is proved in the last subsection 8.3.

8.1 Projection on Configurations

In this subsection, we want to define formally $\eta' \in \text{Disc}(Q_{\text{conf}})$ that would be the result of $\eta \in \text{Disc}(Q_{\text{conf}})$ "deprived of an automaton $\mathcal{A}$." This is achieved in definition 8.4. This definition requires particular precautions and motivate the next sequence of definitions, from definition 8.1 to 8.3.

The next definition captures the idea of a state deprived of a PSIOA $\mathcal{A}$.

**Definition 8.1 (State projection).** Let $A = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be a set of PSIOA partially-compatible at state $q = (q_1, \ldots, q_n) \in Q_{\mathcal{A}_1} \times \ldots \times Q_{\mathcal{A}_n}$. Let $A^s = \{\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_r}\} \subset A$. We note:

- $q \setminus \{\mathcal{A}_k\} = (q_1, \ldots, q_{k-1}, q_{k+1}, \ldots, q_n)$ if $\mathcal{A}_k \in A$ and $q \setminus \{\mathcal{A}_k\} = q$ otherwise.
- $q \setminus A^s = (q \setminus \{\mathcal{A}_{i_1}\}) \setminus (A^s \setminus \{\mathcal{A}_{i_1}\})$ (recursive extension of the previous item).
- $q \uparrow \mathcal{A}_k = q_k$ if $\mathcal{A}_k \in A$ only.
- $q \uparrow A^s = q \setminus (A \setminus A^s)$ (recursive extension of the previous item).

\[ A - (A_1, A_2, A_3, A_4, A_5) \]

\[ A_s - (A_2, A_4) \]

\[ q = (q_1, q_2, q_3, q_4, q_5) \]

\[ q \uparrow A_s = (q_2, q_4) \]

\[ q \setminus A_s = (q_1, q_3, q_5) \]

**Fig. 19. State projection**

The next definition captures the idea of a family transition deprived of a PSIOA $\mathcal{A}$.

**Definition 8.2 (Family transition projection).** (see figure 20 first for an intuition) Let $A = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ be a set of PSIOA partially-compatible at state $q = (q_1, \ldots, q_n) \in Q_{\mathcal{A}_1} \times \ldots \times Q_{\mathcal{A}_n}$. Let $A^s = \{\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_r}\} \subset A$. Let $q' = q \setminus A^s$ and $q'' = q \uparrow A^s$ if $A^s \subset A$. Let $A' = A \setminus A^s$ and $A'' = A^s \subset A$. Let $a' = \text{sig}(A')(q')$ and $a'' = \text{sig}(A'')(q'')$. We note

- $\eta(A, q, a') \setminus A^s \equiv \eta(A', q', a')$ and
- $\eta(A, q, a'') \uparrow A^s \equiv \eta(A'', q'', a'')$ if $A^s \subset A$.

Then we apply this notation to preserving distributions.

**Definition 8.3 (preserving distribution projection).** (see figure 21) Let $\eta_p \in \text{Disc}(Q_{\text{conf}})$ be a preserving distribution. Let $A = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ and its automata support (that is $\forall (A', S') \in \text{supp}(\eta_p), A' = A$). Let $H$ be its set of companion
We note $\eta$ will allow the constructive definition 8.11 of PCA deprived of a (sub) PSIOA. All of this are some versions of law of total probability. The lemma 8.7 and 8.10 (obtained via lemma 8.8), probabilistic measure on configurations generated by $\varphi$-distribution set.

Let $\eta$ be a preserving distribution with $\eta = (\eta_1, ..., \eta_n)$ with $\eta_i \in \text{Disc} (Q_{\mathcal{A}_i})$. Then $\eta_p \downarrow A^s$ is the preserving distribution with $A \downarrow A^s$ as automata support and $H^s = \{ \eta \downarrow A^s | \eta \in H \}$ as companion distribution set.

Now we are able to define intrinsic transition deprived of a PSIOA $\mathcal{R}$.

**Definition 8.4 (intrinsic transition projection).** (see figure 22) Let $\eta \in \text{Disc}(Q_{\text{conf}})$ generated by $\varphi$ and $\eta_p \in \text{Disc}(Q_{\text{conf}})$. We note $\eta \downarrow A^s$ the probabilistic measure on configurations generated by $\varphi \downarrow A^s$ and $\eta_p \downarrow A^s$ and we note $\eta \uparrow A^s$ the probabilistic measure on configurations generated by $\varphi \uparrow A^s$ and $\eta_p \uparrow A^s$.

Then we can easily determine some results when projection is applied. The next lemma 8.5 and 8.6 will lead to lemma 8.7. All of this 3 lemma are some versions of law of total probability. The lemma 8.7 and 8.10 (obtained via lemma 8.8), will allow the constructive definition 8.11 of PCA deprived of a (sub) PSIOA.

**Lemma 8.5 (family distribution projection).** (see figure 23) Let $A = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a finite set of automata. Let $A^s = A \setminus \{\mathcal{A}_k\}$. Let $\eta = \eta_1 \otimes ... \otimes \eta_n$ with $\eta_i \in \text{Disc}(Q_{\mathcal{A}_i})$ for every $i \in [1, n]$. Let $\eta^s = \eta \setminus \{\mathcal{A}_k\}$.

For every $q' \in Q_{\mathcal{A}^s}$, $\eta^s(q') = \sum_{q \in Q_{\mathcal{A}_k}, \eta(k) = q'} \eta(q)$

**Proof.** This comes directly from the law of total probability. We have $\forall q' = (q_1, ..., q_{k-1}, q_{k+1}, ..., q_n) \in Q_{\mathcal{A}_i}, \eta(q') = \eta^s(q') \eta_k(q_k')$. Hence $\sum_{q'_k \in \text{supp}(\eta_k)} \eta(q'_k) = \sum_{q'_k \in \text{supp}(\eta_k)} \eta^s(q'_k) \eta_k(q_k')$, which gives $\sum_{q \in \text{supp}(\eta), q = q' \setminus \{\mathcal{A}_k\}} \eta(q) = \eta^s(q') \cdot \sum_{q'_k \in \text{supp}(\eta_k)} \eta_k(q_k')$ and finally $\sum_{q \in Q_{\mathcal{A}^s}, q = q' \setminus \{\mathcal{A}_k\}} \eta(q) = \eta^s(q')$.

**Lemma 8.6 (preserving distribution projection).** (see figure 24) Let $\eta_p$ be a preserving distribution with $A = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ as automata support. Let $C_Y$ be a configuration $(\eta_p \setminus \{\mathcal{A}_k\})(C_Y) = \sum_{C_X \subseteq C_X \setminus \{\mathcal{A}_k\}} C_Y \eta_p(C_X)$.
Proof. We can apply lemma 8.5 for every pair \((\eta, \eta \setminus \{A_k\})\) s.t. \(\eta\) is a companion distribution of \(\eta_P\) (and \(\eta \setminus \{A_k\}\) is a companion distribution of \(\eta_P \setminus \{A_k\}\) by definition). Then we substitute in the sum of 8.5 every state \(q\) by the corresponding configuration.

\[
\text{LEMMA 8.7 (REDUCED DISTRIBUTION PROJECTION). Let } \eta_P \text{ be a preserving distribution with } A = \{A_1, \ldots, A_n\} \text{ as automata support. Let } \eta_P \text{ be generated by } \varphi \text{ and } \eta_P. \text{ Let } C_Y \text{ be a configuration.}
\]

57
Fig. 23. total probability law for family transition projection

Fig. 24. total probability law for preserving configuration distribution and its companion distribution
\((\eta_r \setminus \{A_1\})(C_Y) = \Sigma_{(C_X, C_X \setminus \{A_1\} = C_Y)} \eta_r(C_X)\).

**Proof.** For a preserving transition, we get \((\eta_p \setminus \{A_1\})(C_Y) = \Sigma_{(C_X, C_X \setminus \{A_1\} = C_Y)} \eta_p(C_X)\) for every configuration \(C_Y\) from Lemma 8.6. By definition 5.10, it follows the same relation for the non-reduced transition which is matching the preserving transition. It follows the same relation for the reduced transition which is matching the non-reduced transition. \(\square\)

The next lemma gives the intrinsic transition attached to a configuration after deprivation of a (sub) PSIOA.

**Lemma 8.8 (Projection on an intrinsic transition).** Let \(C\) be a configuration, \(P\) an automaton, \(a \in \text{signature}(C \setminus P)\), \(\varphi \in \text{Autids}\) and \(\eta \in \text{Disc}(Q_{\text{config}})\), s. t. \(C \xrightarrow{\varphi} \eta_r\). Then, \(C \setminus \{P\} \xrightarrow{a} (\varphi(\{P\}) \setminus \{\eta_r \setminus \{P\}\})\).

**Proof.** We note \(\text{auts}(C) = A = \{A_1, \ldots, A_n\}\), \(S = \text{auts}(C)\) and \(A = A_1||\ldots||A_n\). We note \(q = (S(A_1), \ldots, S(A_n))\).

Since \(a\) is enabled in \(C \setminus \{P\}\), \((q \setminus \{P\}, a, \eta)\) is a transition of \(\mathcal{A}\) (unique from \(q\) and \(a\) by transition determinism), while \((q, a, \eta \setminus \{P\})\) is a transition of \(\mathcal{A}'\) the automaton issued from the composition of automata in \(A \setminus \{P\}\). This comes from the definition of composition 4.10. Now \(\eta_r\) is generated from \(\varphi\) and \(\eta_p\) where \(\eta\) is a companion distribution of \(\eta_p\). In the same way, \(\eta_r \setminus \{P\}\) is generated from \(\varphi \setminus \{P\}\) and \(\eta_p \setminus \{P\}\) where \(\eta \setminus \{P\}\) is a companion distribution of \(\eta_p \setminus \{P\}\).

Thus, \(C \setminus \{P\} \xrightarrow{a} (\eta_p \setminus \{P\})\) and then \(C \setminus \{P\} \xrightarrow{a} (\varphi(\{P\}) \setminus \{\eta_r \setminus \{P\}\})\). \(\square\)

In next subsection, this lemma 8.8 will lead to lemma 8.10 which will be a key lemma to allow the constructive definition 8.11 of PCA deprived of a (sub) PSIOA.

**8.2 \(\mathcal{A}\)-fairness assumption, motivated by our definition of PCA deprived from an internal PSIOA: \(X \setminus \{A\}\)**

Here we recall in definition 8.9 the definition 6.13 of a \(\mathcal{A}\)-fair PCA. Then we show lemma 8.10 (via 8.8) that will be used in complement of lemma 8.7 to enable the constructive definition of \(X \setminus \{A\}\).

**Definition 8.9 (\(\mathcal{A}\)-fair PCA (recall)).** Let \(\mathcal{A} \in \text{Autids}\). Let \(X\) be a PCA. We say that \(X\) is \(\mathcal{A}\)-fair if it verifies the following constraints.

- (configuration-conflict-free) \(X\) is configuration-conflict-free, that is \(\forall q_X, q'_X \in \text{states}(X),\ s.\ t.\ q_X R_{\text{config}} q'_X\) (i.e. \(\text{config}(X)(q_X) = \text{config}(X)(q'_X)\)) then \(q_X = q'_X\).
- (no conflict for projection) \(\forall q_X, q'_X \in \text{states}(X),\ s.\ t.\ q_X R_{\text{config}} q'_X\) then \(q_X R_{\text{strict}} q'_X\). That is if \(\text{config}(X)(q_X) \setminus \{A\} = \text{config}(Y)(q_Y) \setminus \{A\}\), then
  - \(\forall a \in \text{signature}(X)(q_X) \cap \text{signature}(Y)(q_Y), \text{created}(Y)(q_Y)(a) \setminus \{A\} = \text{created}(X)(q_X)(a) \setminus \{A\}\)
  - hidden-actions(X)(q_X) \setminus \text{pot-out}(X)(q_X)(A) = hidden-actions(Y)(q_Y) \setminus \text{pot-out}(Y)(q_Y)(A)\) where
    - \(\text{pot-out}(X)(q)(A) = 0\) if \(A \notin \text{auts}(\text{config}(X)(q))\)
    - \(\text{pot-out}(X)(q)(A) = \text{out}(A)(\text{map}(\text{config}(X)(q)))(A)\) if \(A \in \text{auts}(\text{config}(X)(q))\)
  - (no exclusive creation by \(\mathcal{A}\)) \(\forall q_X \in \text{states}(X), \forall a \in \text{signature}(X)(q_X)\) \(\mathcal{A}\)-exclusive in \(q_X\), \(\text{created}(X)(q_X)(a) = 0\) where \(\mathcal{A}\)-exclusive means \(\forall B \in \text{auts}(\text{config}(X)(q_X)), a \notin \text{signature}(\text{map}(\text{config}(X)(q_X))(B))\).

A \(\mathcal{A}\)-fair PCA is a PCA s. t. we can deduce its current properties from its current configuration deprived of \(\mathcal{A}\). This will allow the definition of \(X \setminus \{A\}\), where \(X\) is a PCA, to be well-defined.
Now we give the second key lemma (after lemma 8.7) to allow the definition 8.11 of PCA deprived of a (sub) PSIOA. Basically, this lemma that if two states $q_X$ and $q_Y$ are strictly equivalent modulo the deprivation of a (sub) automaton $P$, noted $\mathsf{q}_X^{\mathsf{P}}\mathsf{stric}q_Y$, then the intrinsic configurations issued from these states deprived of $P$ are equal.

**Lemma 8.10 (equality of intrinsic transition after deprivation of a sub-PSIOA).** Let $X, Y$ be two PCA. Let $(q_X, q_Y) \in \text{states}(X) \times \text{states}(Y)$ s. t. $q_X^{\mathsf{P}}\mathsf{stric}q_Y$. Let $a \in \text{sig}(X)(q_X) \cap \text{sig}(Y)(q_Y) \setminus (\text{pot-out}(X)(q_X) \cup \text{pot-out}(Y)(q_Y))$. We note $C_X \triangleq \text{config}(X)(q_X), C_Y \triangleq \text{config}(Y)(q_Y), \varphi_X \triangleq \text{created}(X)(q_X)(a), \varphi_Y \triangleq \text{created}(Y)(q_Y)(a)$ and $\eta_X$ and $\eta_Y$ the unique reduced configuration distribution s. t. $C_X \overset{a}{\Rightarrow}_{\varphi_X} \eta_X$ and $C_Y \overset{a}{\Rightarrow}_{\varphi_Y} \eta_Y$. By definition of $R_X^{(P)}stric$, we have $C_X \setminus \{P\} = C_Y \setminus \{P\} \Rightarrow C$ and $\varphi_X \setminus \{P\} = \varphi_Y \setminus \{P\} = \varphi$.

Moreover $C \overset{a}{\Rightarrow}_{\varphi} \eta_r$ with $\eta^X_r \setminus \{P\} = \eta^Y_r \setminus \{P\}$. By unicity of intrinsic transition, we have $\eta^X_r \setminus \{P\} = \eta^Y_r \setminus \{P\}$.

**Proof.** By lemma 8.8, we have both $C \overset{a}{\Rightarrow}_{\varphi} \eta^X_r \setminus \{P\}$ and $C \overset{a}{\Rightarrow}_{\varphi} \eta^Y_r \setminus \{P\}$. By unicity of intrinsic transition, we have $\eta^X_r \setminus \{P\} = \eta^Y_r \setminus \{P\}$.

**Definition 8.11 ($X \setminus \{P\}$).** (see figure 25 for the constructive definition and figures 26 and 27 for the desired result.)

Let $P \in \text{Autids}$. Let $X$ be a $P$-fair PCA, with $\text{psioa}(X) = (Q_X, q_X^0, \text{sig}(X), D_X)$. We note $X \setminus \{P\}$ the automaton $Y$ equipped with the same attributes than a PCA (psioa, config, hidden-actions, created), $\mu^P_X : Q_X \rightarrow Q_Y$ and $\mu^P_d : D_X \setminus \{\eta(X,q_X,a)\} \rightarrow D_Y$ that systematically the following rules:

- $P$-deprivation: $\forall q_Y \in \text{states}(Y), P \notin \text{config}(Y)(q_Y)$, then $\varphi_X(q_X, q_Y) = q_Y$.
- $\mu^P_X$-correspondence: $\forall q_X, q_Y \in Q_X \times Q_Y$ s. t. $\mu^P_X(q_X) = q_Y$, then $\varphi_X(q_X) = q_Y$.
- $\mu^P_d$-correspondence: $\forall q_X, q_Y \in Q_X \times Q_Y$, $\varphi_X(q_X, a) = \varphi_Y(q_Y, a)$ s. t. $\eta_Y(a, q_Y) = \mu^P_d(\eta(X,q_X,a))$.

and constructed (conjointly with the mapping $\mu^P_d$ as follows:

- Partitioning: We partition $Q_X$ in equivalence classes according to the equivalence relation $R_X^{(P)}_{\text{conf}}$ that is we obtain a partition $(C_{q_X}^j)_{j \in \mathbb{N}}$ s. t. $\forall q_X \in C_{q_X}^j$, $q_X \overset{a}{\Rightarrow}_{\varphi} \eta_{q_X}$ and by $P$-fair assumption, $q_X \overset{a}{\Rightarrow}_{\varphi} \eta_{q_X}$. We note $C_{q_X}^j \setminus \{P\} = C_{r_X}^j$.

Moreover $\varphi_Y(q_Y) = \mu^P_d(\varphi_X(q_X))$.

- $\varphi_Y(q_Y) = \mu^P_d(\varphi_X(q_X))$ (and so $\forall q_X \in (\mu^P_d)^{-1}(q_Y), a \in \text{sig}(X)(q_X)$) we construct $\eta_Y(a, q_Y) = \mu^P_d(\eta(X,q_X,a))$ and conjointly extend $\mu^P_d$ s. t. $\forall q_X \in (\mu^P_d)^{-1}(q_Y), \eta_Y(q_Y,a) = \mu^P_d(\eta(X,q_X,a))$, verifying the $\mu^P_d$-correspondence rule.

We show this construction is possible:

- We note $C_Y \triangleq \text{config}(Y)(q_Y), \varphi_Y \triangleq \text{created}(Y)(q_Y)(a)$, $\eta_Y$ the unique reduced configuration distribution so that $C_Y \overset{a}{\Rightarrow}_{\varphi_Y} \eta_Y$. Let $(\eta^j_{q_X})_{j \in \mathbb{N}} = (\mu^P_d)^{-1}(q_Y)$. For every $i \in I$, we note $C^i_X = \text{config}(X)(q^i_X)$, $\varphi^i_X \triangleq \text{created}(X)(q^i_X)(a)$, $\eta^i_X$, the unique reduced configuration distribution so that $C^i_X \overset{a}{\Rightarrow}_{\varphi^i_X} \eta^i_X$. For every $i \in I$, we have $C_X \setminus \{P\} = C_Y, \varphi^i \setminus \{P\} = \varphi_Y$ and $\eta^i_X \setminus \{P\} = \eta_Y$ by lemma 8.10.

For every $q^i_X \in (\mu^P_d)^{-1}(q_Y)$, we partition $\text{supp}(\eta(X,q^i_X,a))$ in equivalence classes according to the equivalence relation $R_X^{(P)}_{\text{conf}}$ that is we obtain a partition $(C^j_{q^i_X})_{j \in \mathbb{N}}$ s. t. $\forall j \in J, \forall q^j_{X^j} \in C^j_{q^i_X} \setminus \{P\}$ and
by $P$-fair assumption, $q'_X \setminus \text{strict}q'_Y$. For each $j \in J'$, we extract an arbitrary $q'_X \in C'_j$ and $q'_Y = \mu'_j(q'_X)$. We fix $\eta_{(Y,a)}(q'_Y) = \eta_Y(\text{config}(Y)(q'_Y))$.

Now by lemma 8.1, $\eta_Y(C'_Y) = \sum C'_X C'_Y = C'_X \setminus \{P\} \eta'_X(C'_X)$. By constraint 3 of bottom/up transition preservation, $\sum C'_X C'_Y = C'_X \setminus \{P\} \eta'_X(C'_X) = \sum q'_X q'_Y = \mu'_j(q'_X) \eta_X(q'_X) = \sum q'_X q'_Y = \mu'_j(q'_X) \eta_X(q'_X) = \eta_X(q'_X)$. Thus, the $\mu'_j$ correspondence constraint, i.e. $\eta_{(Y,a)}(q'_Y) = \sum q'_X q'_Y = \mu'_j(q'_X) \eta_X(q'_X) = \eta_X(q'_X)$ holds for all the possible $q'_X \in (\mu'_j)^{-1}(q'_Y)$.

Fig. 25. constructive definition of $Y = X \setminus \{P\}$. First we construct $\tilde{q}_0$ which is the initial state of $Y$. Then we partition $\text{supp}(\eta_X(a)) = \{q^{1x}_X, q^{1y}_X\} \cup \{q^{1x}_Y, q^{1y}_Y\}$ s.t. $q^{1x}_X \in \text{supp}(\eta_X(a))$ and $q^{1y}_X \in \text{supp}(\eta_X(a))$. Then, $\eta_Y(q_0)$ is defined s.t. $\eta_Y(q_0) = \eta_X(q_0) \neq \eta_X(q_0)$. We perform another time this procedure by partitioning $\text{supp}(\eta_X(q_0)) = \{q^{2x}_X \cup \{q^{2y}_Y\}$ or $\text{supp}(\eta_X(q_0)) = \{q^{2x}_X, q^{2y}_X \cup \{q^{2y}_Y, q^{2y}_Y\}$ arbitrarily. Indeed the obtained result is the same: $q^{1x}_X \in \text{supp}(\eta_X(q_0))$ since they are both pre-image of $q^{1y}_Y$ by $\mu$, which means $\{q^{1y}_X \in \text{supp}(\eta_X(q_0)) \}$. If we note $\text{config}(X)(q^{1y}_X)$, $\text{config}(X)(q^{1y}_Y)$, $\phi_u = \text{create}(X)(q^{1y}_X)$, $\phi_v = \text{create}(X)(q^{1y}_Y)$, $\phi_u \equiv \phi_v$, $\eta_u \equiv \phi_u$, and $\eta_v \equiv \phi_v$, we have $j)$ $C_u \setminus \{P\} \equiv \phi_u \mid \{P\}$, $\eta_u \equiv \phi_u \mid \{P\}$ and $j)$ $C_u \setminus \{P\} \equiv \phi_v \mid \{P\}$, $\eta_v \equiv \phi_v \mid \{P\}$ which implies $jv)$ $\eta_u \equiv \phi_u \mid \{P\}$.

In the remaining, if we consider a PCA $X$ deprived of a PSIOA $\mathcal{A}$ we always implicitly assume that $X$ is $\mathcal{A}$-fair.
8.3 $Y = X \setminus \{A\}$ is a PCA if $X$ is $\mathcal{A}$-fair

Here we prove a sequence of lemma to show that $Y = X \setminus \{A\}$ is indeed a PCA, by verifying all the constraints.

Prepare the top/down transition preservation. We show a useful lemma to show $Y = X \{A\}$ verifies the constraint 2 of top/down transition preservation.

**Lemma 8.12 (corresponding transition after projection).** Let $\mathcal{A}$ be a PSIOA. Let $X$ be a $\mathcal{A}$-fair PCA. Let $Y = X \setminus \{A\}$. Let $(q_x, a, \eta_x) \in \text{dtrans}(X)$ and $a \in \text{sig}(\text{config}(X)|q_x\setminus\{P\})$.

Let $\eta'_X \in \text{Disc}(\text{Conf})$ the unique reduced configuration transition s. t. $x0) \eta(X,q_x,a) \xrightarrow{\text{f}} \eta'_X$ with $x1) \text{f} = \text{config}(X)(q_X)$ and $x2) \text{Config}(X)(q_X) \xrightarrow{\text{a}} \varphi_X \eta'_X$ where $\varphi_X = \text{created}(X)(q_X)(a)$.

Let $(q_y, a, \eta(Y,q_Y,a))$ with $\eta(Y,q_Y,a) = \mu_q(\eta_X))$, $q_y = \mu_q(q_x)$. Let $\eta'_Y = \eta'_X \setminus \{A\}$ Then $\eta'_Y$ is a reduced configuration transition that verifies $y0) \eta(Y,q_Y,a) \xrightarrow{\text{f'}} \eta'_Y$ with $y1) \text{f'} = \text{config}(Y)(q_Y)$ and $y2) \text{Config}(Y)(q_Y) \xrightarrow{\text{a}} \varphi_Y \eta'_Y$ where $\varphi_Y = \varphi_X \setminus \{A\} = \text{created}(Y)(q_Y)(a)$.

**Proof.** We note $(Q^X_i)_{i \in I}$ the partition of $\text{supp}(\eta_X|q_X,a)$ s. t. $\forall i \in I \forall q''_X, q''_X \in Q^X_i, q''_X \xrightarrow{\text{r}} q''_X \forall i \in I$, we note $Q^X_i|\{\mathcal{A}\} = \text{config}(q''_X) \setminus \{\mathcal{A}\}$ for an arbitrary element $q''_X \in Q^X_i$ and $C_i = \{C \in \text{supp}(\eta_X)|C \setminus \mathcal{A} = C^1_i \setminus \{\mathcal{A}\} \}$. Since $x0)$ $\eta(X,q_X,a) \xrightarrow{\text{f}} \eta'_X$ with $x1) \text{f} = \text{config}(X)(q_X)$, $(C_i)_{i \in I}$ is a partition of $\text{supp}(\eta_X)$.
For every \( i \in I \), we note \( q_i^Y = \mu_s(q_i^X) \) for an arbitrary element \( q_i^X \in Q_i^X \). By \( \mu_s^A \)-correspondance, \( \text{config}(q_i^Y) = C_i^A \) = \( \text{config}(q_i^X) \) \( \setminus \{A\} \).

By \( \mu_s^A \)-correspondance, \( \eta_{Y,A,s}(q_i^Y) = (\mu_s(\eta(X,q_X,s)))(q_i^Y) = \Sigma_{q,s}(\eta(\mu_s(q_X,s))=q_i^Y \text{ config}(X)(q_X)) = \Sigma_{i \in I} \eta_X(q_i^X) = q_i^Y \eta(X,q_X,s) \).

By assumption x0) and x1), \( \eta(X,q_X,s) \) \( \mapsto \eta_X' \) with \( f = \text{config}(X) \), thus \( \eta_{Y,A,s}(q_i^Y) = \Sigma_{i \in I} \eta_X(q_i^X) = q_i^Y \eta_X(\text{config}(X)(q_i^Y)) = \Sigma_{i \in I} \eta_X(q_i^X) \).

Thereafter, we use the lemma 8.7 and get \( \eta_{Y,q_X,s}(q_i^Y) = \eta_Y(\text{config}(X)(q_i^Y)) \) with \( \eta_X' = \eta_X^A \setminus \{A\} \).

By definition of \( Y \), \( \text{config}(Y)(q_Y) = \mu_s(q_Y) \) = \( \text{config}(X)(q_X) \setminus \{A\} \). Then, since \( a \in \text{supp}(\text{config}(X)(q_X) \setminus \{A\}) \), we can apply lemma 8.8. Thus \( \text{config}(Y)(q_Y) \xrightarrow{a} \varphi_Y \eta_Y' \) with \( \eta_Y' = \eta_Y^A \setminus \{A\} \) and \( \varphi_Y = (q_X \setminus \{A\}) \). By \( \mu_s^\mathcal{A} \)-correspondance, \( \text{created}(Y)(q_Y)(a) = \text{created}(X)(q_X)(a) \setminus \{A\} \), thus \( \varphi_Y = \text{created}(Y)(q_Y)(a) \).

Finally the restriction of \( \text{config}(Y) \) on \( \text{supp}(\eta_Y,q_Y,s) \) is a bijection. Indeed, we note \( f_1 : q_Y \mapsto Q_i^X \) s. t. \( q_Y = \mu_s(Q_i^X) \), \( f_2 : Q_i^X \mapsto C_i f_3 : C_i \mapsto C_i^A \). By construction, \( f_1 \) and \( f_3 \) are bijection. By bijectivity of the restriction of \( \text{config}(X) \) on \( \text{supp}(\eta_X,q_X,s) \), \( f_2 \) is a bijection too. Moreover, the restriction \( f' \) of \( \text{config}(Y) \) on \( \text{supp}(\eta_Y,q_Y,s) \) is \( f_1 \circ f_2 \circ f_3 \) and hence this is a bijection too.

\( \square \)

We show a useful lemma to show \( Y = X \setminus \{A\} \) verifies the constraint 3 of bottom/up transition preservation.
LEMMA 8.13 (existence of intrinsic transition). Let \( \mathcal{A}_k \in \text{Autids} \), let \( \mathcal{X} \) be a \( \mathcal{A}_k \)-fair PCA, let \( Y = X \setminus \{ \mathcal{A}_k \} \) and \( q_Y \in \text{States}(Y) \).

If \( \exists \eta'_X \in \text{Disc}(Q_{\text{conf}}), \eta' \in \text{Download}(\text{Config}(Y)(q_Y)), \text{Config}(Y)(q_Y)(a) \) s. t. \( \text{Config}(Y)(q_Y) \Rightarrow a \eta' \), then it exists \( \exists X \in \text{States}(X), \mu_X(q_Y) = q_Y, \eta'_X \in \text{Disc}(Q_{\text{conf}}), \eta'_X = (\eta' \setminus \{ \mathcal{A}_k \}), a \in \text{Download}(\text{Config}(X)(q_X) \setminus \{ \mathcal{A}_k \}), \text{Config}(X)(q_X) \setminus \{ \mathcal{A}_k \} \).

PROOF. By construction of \( Y = X \setminus \{ \mathcal{A}_k \} \), if \( q_Y \in \text{states}(Y) \), it exists \( q_X \in \text{states}(X), \mu_X(q_Y) = q_Y, \text{config}(X)(q_X) \setminus \{ \mathcal{A}_k \} = \text{config}(Y)(q_Y) \) and \( \text{created}(X)(q_X)(a) = \text{created}(Y)(q_Y)(a) \setminus \{ \mathcal{A}_k \} \). We note \( C_X = (A_X, S_X) = \text{config}(X)(q_X) \) and \( C_Y = (A_Y, S_Y) = \text{config}(Y)(q_Y) \). We treat two cases to show that \( \eta(C_Y, a, p) = \eta(C_X, a, p) \setminus \{ \mathcal{A}_k \} \):

- case 1) \( \mathcal{A}_k \in A_X \) with \( S_X(\mathcal{A}_k) = q_k \), we note \( (A_X = A_{1, \ldots, k, \ldots, n}) \). Thus \( (A_Y = A_{1, \ldots, k-1, k-1, \ldots, n}) \).

For every \( i \in [1:n] \), we note \( \eta^i = \eta(C_i, q_i, a) \) if \( a \in \text{Download}(\mathcal{A}_i)(q_i) \) and \( \eta^i = \delta_{q_i} \).

By definition of companion distribution, \( C_Y \Rightarrow a \eta(C_Y, a, p) \) and \( C_X \Rightarrow a \eta(C_X, a, p) \), where \( \eta(C_X, a, p) \) has the companion distribution \( \eta_X^p = \eta^1 \otimes \ldots \otimes \eta^{k-1} \otimes \eta^{k+1} \otimes \ldots \otimes \eta^n \).

Hence \( \eta(C_Y, a, p) = \eta(C_X, a, p) \setminus \{ \mathcal{A}_k \} \).

- case 2) \( \mathcal{A}_k \notin A_X \). Immediate, since we have \( A_X = A_Y \) and \( \mathcal{A}_k \notin A_X \).

Let us note \( \varphi_X = \text{created}(X)(q_X)(a) \) and \( \varphi_Y = \text{created}(Y)(q_Y)(a) = \varphi_Y \setminus \{ \mathcal{A}_k \} \). We note \( \eta'_X \) the reduced configuration distribution generated by \( \varphi_X \) and \( \eta(C_X, a, p) \) and by definition 5.11 of intrinsic transition, we have \( C_X \Rightarrow a \eta'_X \).

By definition 5.11 of intrinsic transition, \( C_Y \Rightarrow a \eta_Y^p \) with \( \eta'_Y \) generated by \( \eta(C_Y, a, p) = \eta(C_X, a, p) \setminus \{ \mathcal{A}_k \} \) and \( \varphi_Y = \varphi_X \setminus \{ \mathcal{A}_k \} \).

Thus, \( \eta'_X = \eta'_Y \setminus \{ \mathcal{A}_k \} \) by definition 8.4 of intrinsic transition projection, which ends the proof.

\[ \square \]

Now we are able to demonstrate that the PCA set is closed under deprivation.

THEOREM 8.14 \( (X \setminus \{ P \}) \) is a PCA. Let \( P \in \text{Autids} \). Let \( X \) be a \( P \)-fair PCA, then \( Y = X \setminus \{ P \} \) is a PCA.

PROOF.

- (Constraint 1) By construction of \( Y, \tilde{q}_Y = \mu_{X, P}(\tilde{q}_X) \) and by \( \mu_{X, P} \)-correspondence rule, \( \text{config}(Y)(\tilde{q}_Y) = \text{config}(X)(\tilde{q}_X) \setminus \{ P \} \). Since the constraint 1 is respected by \( X \), it is a fortiori respected by \( Y \).

- (Constraint 2) Let \( q_Y, a, \eta(Y, q_Y, a) \in \text{dtrans}(Y) \). By construction of \( Y \), we know it exists \( \eta(X, q_X, a) \in \text{dtrans}(X) \) with \( \eta(X, q_X, a) = \mu_{X, P}(\eta(X, q_X, a)) \) and \( q_Y = \mu_X(q_X) \). Then, because of constraint 2 ensured by \( X \), we obtain it exists a reduced configuration distribution \( \eta'_X \in \text{Disc}(Q_{\text{conf}}) \) s. t. \( \eta'_X \setminus \{ P \} \) exists.

- (Constraint 3) Let \( q_Y \in \text{States}(Y), \eta'_X \in \text{Disc}(Q_{\text{conf}}), a \in \text{Download}(\text{Config}(Y)(q_Y)), \varphi_Y = \text{created}(Y)(q_Y)(a) \) s. t. \( \text{Config}(Y)(q_Y) \Rightarrow a \eta'_Y \).

Because of Lemma 8.13, it implies it exists \( \varphi_X \in \text{States}(X), \mu_X(q_Y) = q_Y, \) s. t. \( \text{config}(X)(q_X) \Rightarrow a \eta'_X \) with \( \varphi_X = \text{created}(X)(q_X)(a), \eta'_X = \eta'_X \setminus \{ P \}, \varphi_Y = \varphi_X \setminus \{ P \} \).
Because of constraint 3, it means \((q_X, a, \eta_{X,q_X,a}) \in d\text{trans}(X)\) with \(x_0) \eta_{(X,q_X,a)} \xrightarrow{f} \eta'_{X} \) with \(x_1) f = \text{config}(X)(q_X).\) Since \(q_Y = \mu_X(q_X)\) and \(a \in \text{sig}(Y)(q_Y)\), the construction of \(d\text{trans}(Y)\) implies \((q_Y, a, \eta_{(Y,q_Y,a)}) \in d\text{trans}(Y)\) with \(\eta_{(Y,q_Y,a)} = \mu_{Y}^{a}(\eta_{(X,q_X,a)})\).

We can reapply lemma 9.4.1. To obtain that \(\eta''_Y = \eta'_X \setminus \{P\}\) is a reduced configuration transition that verifies 
\(y(0) \eta_{(Y,q_Y,a)} \xrightarrow{f''} \eta''_Y\) with \(y(1) f'' = \text{config}(Y)(q_Y)\). Then \(\text{config}(Y)(q_Y) \xrightarrow{a} \eta''_Y\) where \(\phi_Y = \eta_X \setminus \{P\}\).

Finally, \(\eta''_Y = \eta'_X \setminus \{P\}\), which allows us to conclude that 
For every \(q_Y \in \text{States}(Y)\), \(a \in \text{sig}(\text{config}(Y)(q_Y))\), \(\phi_Y = \text{config}(Y)(q_Y)\) s. t. \(\text{config}(Y)(q_Y) \xrightarrow{a} \eta''_Y\). with some reduced configuration distribution \(\eta'_Y\), then \((q_Y, a, \eta_{(Y,q_Y,a)}) \in d\text{trans}(Y)\) with \(\eta_{(Y,q_Y,a)} \xrightarrow{f'} \eta''_Y\)
where \(f' = \text{config}(Y)(q_Y)\). This terminates the proof of constraint 3.

- (Constraint 4) Verified by construction. (We recall that \(Y(q_X, q_Y) \in \text{States}(Y) \times \text{States}(X)\), \(q_Y = \mu_Y^{(X)}(q_X), \text{sig}(Y)(q_Y) \triangleq \text{hide}(\text{config}(Y)(q_Y), \text{hidden-acts}(Y)(q_Y))\) where \(\text{hidden-acts}(Y)(q_Y) \triangleq \text{hidden-acts}(X)(q_X) \setminus \text{pot-out}(X)(q_X)(P)\).

\[\square\]

9 RECONSTRUCTION

In the previous section, we have shown that \(Y = X \setminus A\) is a PCA (as long as \(X\) is \(A\)-fair). In this section we will

1. introduce the concept of simpleton wrapper \(\hat{A}^{sw}\) that is a PCA that encapsulates \(A\).
2. prove that \(X \setminus \{A\}\) and \(\hat{A}^{sw}\) are partially-compatible (see theorem 9.13).
3. There is a strong executions-matching from \(X\) to \((X \setminus \{A\})||\hat{A}^{sw}\) in a restricted set of executions of \(X\) that do not create \(A\) (see theorem 9.19). Hence it is always possible to transfer a reasoning on \(X\) into a reasoning on \((X \setminus \{A\})||\hat{A}^{sw}\) if no re-creation of \(A\) occurs.
4. The operation of projection/deprivation and composition are commutative (see theorem 9.24).

9.1 Simpleton wrapper \(\hat{A}^{sw}\)

Here we introduce simpleton wrapper \(\hat{A}^{sw}\), a PCA that only encapsulates \(\hat{A}^{sw}\).

**Definition 9.1 (Simpleton wrapper).** (see figure 28) Let \(A\) be a PSIOA. We note \(\hat{A}^{sw}\) the simpleton wrapper of \(A\) as the following PCA:

- It exists a bijection \(\text{ren}_{sw} : Q_A \rightarrow Q_{\hat{A}^{sw}}\) \(s.t.\) \(\text{psioa}(\hat{A}^{sw}) = \text{ren}_{sw}(A)\), that is \(\text{psioa}(\hat{A}^{sw})\) differs from \(A\) only syntactically.
- \(\forall q_{\hat{A}^{sw}} \in \text{states}(\hat{A}^{sw}), \text{config}(\hat{A}^{sw})(\hat{q}_{\hat{A}^{sw}}) = \text{reduced}(\{A\}), S : A \mapsto q_A \Rightarrow q_A \neq \text{ren}_{sw}^{-1}(q_{\hat{A}^{sw}}(A))\)
- \(\forall q_{\hat{A}^{sw}} \in \text{states}(\hat{A}^{sw}), \forall a \in \text{sig}(\hat{A}^{sw})(\hat{q}_{\hat{A}^{sw}}), \text{hidden-acts}(\hat{A}^{sw})(\hat{q}_{\hat{A}^{sw}}) = \emptyset\) and \(\text{created}(\hat{A}^{sw})(\hat{q}_{\hat{A}^{sw}}(a)) = \emptyset\).

We can remark that when \(\hat{A}^{sw}\) enters in \(q_{\hat{A}^{sw}} = \text{ren}_{sw}(\hat{q}_{\hat{A}^{sw}})\) where \(\text{sig}(\hat{A}^{sw})(\hat{q}_{\hat{A}^{sw}}) = \emptyset\), this matches the moment where \(A\) enters in \(q_A\) where \(\text{sig}(A)(q_A) = \emptyset\), s. t. the corresponding configuration is the empty one.
Lemma 9.2. Let $\mathcal{A}$ be a PSIOA. Let $\widetilde{\mathcal{A}}_{sw}$ its simpleton wrapper with $\text{psoia}(\widetilde{\mathcal{A}}_{sw}) = \text{ren}_{sw}(\mathcal{A})$. Let $\mu \in \text{Disc}(\text{Frags}(\widetilde{\mathcal{A}}_{sw}))$.

For every schedule $\rho$, apply $\widetilde{\mathcal{A}}_{sw}(\text{ren}_{sw}(\mu), \rho)(\text{ren}_{sw}(\alpha)) = \text{apply}_A(\mu, \rho)(\alpha)$.

Proof. The only point is that (i) $\forall q \in \text{states}(\mathcal{A})$, $\text{constitution}(\widetilde{\mathcal{A}}_{sw})(\text{ren}_{sw}(q)) = \text{constitution}(\mathcal{A})(q)$ and (ii) for $q^0$ s.t. $\text{sig}(\mathcal{A})(q^0) = \emptyset$, $\text{constitution}(\text{tilde}(\mathcal{A}_{sw}))(\text{ren}_{sw}(q^0)) = \emptyset$ which means that $(*) T$ is enabled in $q$ iff $T$ is enabled in $\text{ren}_{sw}(q)$ and that $(**)$ $a$ is triggered by $T$ in state $q$ iff $a$ is triggered by $T$ in state $\text{ren}_{sw}(q)$.

Thus we can apply theorem 7.25.

\[\Box\]

9.2 Partial-compatibility of $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\widetilde{\mathcal{A}}_{sw}$

In this subsection, we show that $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\widetilde{\mathcal{A}}_{sw}$ are partially-compatible and that $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})|\widetilde{\mathcal{A}}_{sw}$ mimics $X_{\mathcal{A}}$ as long as no creation of $\mathcal{A}$ occurs (see figure 29).
Map $X$ and $(X \setminus \{A\}, \hat{A}^{sw})$. We first introduce two functions to map $X$ and $(X \setminus \{A\}, \hat{A}^{sw})$.

**Definition 9.3 ($\mu_A^X$ and $\mu_A^Y$: mapping of reconstruction).** Let $A \in Autids$, $X$ be a $A$-fair PCA, $Y = X \setminus A$. Let $\hat{A}^{sw}$ be the simpleton wrapper of $A$, where $psioa(\hat{A}^{sw}) = ren_{sw}(A)$. Let $q_A^\phi \in states(A)$ the (assumed) unique state s.t. $\bar{\sigma}(A)(q_A^\phi) = 0$. We note:

- The function $X.\mu_A^X : states(X) \to states(Y) \times states(\hat{A}^{sw})$ s.t. $\forall q_X \in states(X), X.\mu_A^X(q_X) = (X.\mu_A^X(q_X), ren_{sw}(q_A))$ with $q_A = map(config(X)(q_X))(A)$ if $A \in (auts(config(X)(q_X)))$ and $q_A = q_A^\phi$ otherwise.

Fig. 29. Reconstruction of a PCA via $Z = (X, X \setminus \{V\})$
The function \( \mu_\mathcal{A} \) that maps any alternating sequence \( a_X = q_X^0, a, q_X^1, a^2 \ldots \) of states and actions of \( X \), to 
\[ \mu_\mathcal{A}(a_X) \] the alternating sequence \( a_Z = X, \mu_\mathcal{A}(q_X^0), a, X, \mu_\mathcal{A}(q_X^1), a^2, \ldots \).

The symbol \( \mathcal{A} \) and \( X \) are omitted when this is clear in the context.

Now, we recall definition 6.14 of \( \mathcal{A} \)-conservative PCA, an additional condition to allow the compatibility between \( X \setminus \mathcal{A} \) and \( \mathcal{A}^{sw} \).

**Definition 9.4 (\( \mathcal{A} \)-conservative PCA (recall)).** Let \( X \) be a PCA, \( \mathcal{A} \in \text{Autids} \). We say that \( X \) is \( \mathcal{A} \)-conservative if it is \( \mathcal{A} \)-fair and for every state \( q_X \in \text{States}(X) \), \( C_X = (A_X, S_X) = \text{config}(X)(q_X) \) s. t. \( \mathcal{A} \in A_X \) and \( S_X(\mathcal{A}) = q_X \).

The next lemma shows that reconstruction preserves signature compatibility.

**Lemma 9.6 (preservation of signature).** Let \( \mathcal{A} \in \text{Autids} \). Let \( X \) be a \( \mathcal{A} \)-conservative PCA, \( Y = X \setminus \mathcal{A} \). Let \( q_X \in \text{States}(X) \), \( C_X = (A_X, S_X) = \text{config}(X)(q_X) \). Let \( q_Y \in \text{States}(Y) \), \( q_Y = \mu_\mathcal{A}(q_X) \). Let \( C_Y = (A_Y, S_Y) = \text{config}(Y)(q_Y) \).

If \( \mathcal{A} \in A_X \) and \( q_\mathcal{A} = S_X(\mathcal{A}) \), then \( \text{sig}(C_Y) \) and \( \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \) are compatible and \( \text{sig}(C_X) = \text{sig}(C_Y) \times \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \).

If \( \mathcal{A} \notin A_X \), then \( \text{sig}(C_Y) \) and \( \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \) are compatible and \( \text{sig}(C_X) = \text{sig}(C_Y) \times \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \).

**Proof.** Let \( \mathcal{A} \in \text{Autids} \) Let \( X \) and \( Y \setminus \{\mathcal{A}\} \) be PCA. Let \( q_X \in \text{States}(X) \). Let \( C_X = \text{config}(X)(q_X), A_X = \text{auts}(C_X) \) and \( S_X = \text{map}(C_X) \). Let \( q_Y \in \text{States}(Y) \), \( q_Y = \mu_\mathcal{A}(q_X) \). Let \( C_Y = \text{config}(Y)(q_Y) \), \( A_Y = \text{auts}(C_Y) \) and \( S_Y = \text{map}(C_Y) \).

By definition of \( Y \), \( C_Y = C_X \setminus \{\mathcal{A}\} \).

**Case 1: \( \mathcal{A} \in A_X \)**

Since \( X \) is a PCA, \( C_X \) is a compatible configuration, thus \( ((A_Y, S_Y) \cup (\mathcal{A}, q_\mathcal{A})) \) is a compatible configuration. Finally \( \text{sig}(C_Y) \) and \( \text{sig}(\mathcal{A})(q_\mathcal{A}) \) are compatible with \( \text{sig}(\mathcal{A})(q_\mathcal{A}) = \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \).

By definition of intrinsic attributes of a configuration, that are constructed with the attributes of the automaton issued from the composition of the family of automata of the configuration, we have \( A_X = A_Y \cup \{\mathcal{A}\} \) and \( \text{sig}(C_X) = \text{sig}(C_Y) \times \text{sig}(\mathcal{A})(q_\mathcal{A}) \), that is \( \text{sig}(C_X) = \text{sig}(C_Y) \times \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \).

**Case 2: \( \mathcal{A} \notin A_X \)**

Since \( X \) is a PCA, \( C_X \) is a compatible configuration, thus \( C_Y = C_X \) is a compatible configuration. Finally \( \text{sig}(C_Y) \) and \( \text{sig}(\mathcal{A})(q_\mathcal{A}) = (\emptyset, \emptyset, \emptyset) = \text{sig}(\mathcal{A})(q_\mathcal{A}) = \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \) are compatible.

By definition of intrinsic attributes of a configuration, that are constructed with the attributes of the automaton issued from the composition of the family of automata of the configuration (here \( A_Y \) and \( A_X = A_Y \)), we have \( \text{sig}(C_X) = \text{sig}(C_Y) \). Furthermore, \( \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) = \text{sig}(\mathcal{A})(q_\mathcal{A}) = (\emptyset, \emptyset, \emptyset) \). Thus \( \text{sig}(C_X) = \text{sig}(C_Y) \times \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \). 

The next lemma shows that reconstruction preserves signature.

**Lemma 9.7 (preservation of signature).** Let \( \mathcal{A} \in \text{Autids} \). Let \( X \) be a \( \mathcal{A} \)-conservative PCA, \( \mathcal{A} \in \text{Autids}, Y = X \setminus \{\mathcal{A}\} \). For every \( q_X \in \text{States}(X) \), we have \( \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y) \times \text{sig}(\mathcal{A}^{sw}(\text{ren}_{sw}(q_\mathcal{A}))) \) with \( (q_Y, \text{ren}_{sw}(q_\mathcal{A})) = \mu_\mathcal{A}^\mathcal{A}(q_X) \).
Proof. The last lemma 9.5 tell us for every $q_X \in \text{states}(X)$, we have $\text{sig}(\text{config}(X)(q_X)) = \text{sig}(\text{config}(Y)(q_Y)) \times \text{sig}(\tilde{A}^w)(\text{ren}_{sw}(q_A))$ with $(q_Y, \text{ren}_{sw}(q_A)) = \mu_Z(q_X)$. Since $X$ is $\mathcal{A}$-conservative, we have $(\ast) \text{sig}(X)(q_X) = \text{hide}(\text{config}(X)(q_X)), \text{acts}$ where acts $\subseteq (\text{out}(X)(q_X) \setminus (\text{ext}(\mathcal{A})(q_A)))$. Hence $\text{sig}(Y)(q_Y) = \text{hide}(\text{config}(Y)(q_Y)), \text{acts}$. Since $(\ast)$ acts $\cap \text{ext}(\mathcal{A})(q_A) = \emptyset$, $\text{sig}(Y)(q_Y)$ and $\text{sig}(\mathcal{A})(q_A)$ are also compatible. We have $\text{sig}(\text{config}(X)(q_X)) = \text{sig}(\text{config}(Y)(q_Y)) \times \text{sig}(\tilde{A}^w)(\text{ren}_{sw}(q_A))$ which gives because of $(\ast)$ hide($\text{config}(X)(q_X)), \text{acts} = \text{hide}(\text{config}(Y)(q_Y)), \text{acts} \times \text{sig}(\mathcal{A})(q_A)$, that is $\text{sig}(X)(q_X) = \text{sig}(Y)(q_Y) \times \text{sig}(\mathcal{A})(q_A) = \text{sig}(Y)(q_Y) \times \text{sig}(\tilde{A}^w)(\text{ren}_{sw}(q_A))$. □

The next lemma shows that reconstruction preserves partial-compatibility at any reachable state.

Lemma 9.7 (preservation of partial-compatibility at any reachable state). Let $\mathcal{A} \in \text{Autids}$, $X$ be a $\mathcal{A}$-conservative PCA, $Y = X \setminus \{A\}$, $Z = (Y, \tilde{A}^w)$. Let $q_Z = (q_Y, q_{\tilde{A}^w}) \in \text{states}(Y) \times \text{states}(\tilde{A}^w)$ and $q_X \in \text{states}(X)$ s. t. $\mu_Z(q_X) = q_Z$. Then $Z$ is partially compatible at state $q_Z$ (in the sense of definition 5.19).

Proof. Since $X$ is a $\mathcal{A}$-conservative PCA, the previous lemma 9.6 ensures that $\text{sig}(Y)(q_Y)$ and $\text{sig}(\mathcal{A})(q_A) = \text{sig}(\tilde{A}^w)(\text{ren}_{sw}(q_A))$ are compatible, thus by definition $Z$ is partially compatible at state $q_Z$. □

Here, we show that reconstruction preserves probabilistic distribution of corresponding transition, as long as no creation of the concerned automaton occurs.

Lemma 9.8 (homomorphic transition without creation). Let $\mathcal{A} \in \text{Autids}$, $X$ be a $\mathcal{A}$-conservative PCA, $Y = X \setminus \{A\}$, $Z = (Y, \tilde{A}^w)$. Let $q_Z = (q_Y, q_{\tilde{A}^w}) \in \text{states}(Y) \times \text{states}(\tilde{A}^w)$ and $q_X \in \text{states}(X)$ s. t. (i) $\mu_Z(q_X) = q_Z$. Let $a \in \text{sig}(X)(x) = \text{sig}(Y)(y) \times \text{sig}(\tilde{A}^w)(\tilde{q}_{\tilde{A}^w})$, verifying (ii: No creation from $\mathcal{A}$) If $a$ is $\mathcal{A}$-exclusive in state $q_X$, then created$(X)(a) = 0$.

- If $\mathcal{A}$ is not created by $a$, i. e. if either
  - $\mathcal{A} \notin \text{acts}(\text{config}(X)(x))$, or
  - $\mathcal{A} \notin \text{acts}(\text{config}(X)(x)) \text{ and } \mathcal{A} \notin \text{created}(X)(a)$ (X does not create $\mathcal{A}$ with probability 1)

  Then $\eta(X,q_X,a) \xrightarrow{\mu_X} \eta(Z,q_Z,a)$

- If $\mathcal{A}$ is created by a i. e. $\mathcal{A} \notin \text{acts}(\text{config}(X)(x)) \text{ and } \mathcal{A} \notin \text{created}(X)(a)$ (X creates $\mathcal{A}$ with probability 1)

  Then $\eta(X,q_X,a) \xrightarrow{\phi} \eta(Z,q_Z,a)$ where $\phi : q'_X \in \supp(\eta(X,q_X,a)) \mapsto (X,\mu^A_X(q'_X),q^A_{\tilde{A}^w})$.

Proof. By lemma 9.6, we have $\text{sig}(X)(q_X) = \text{sig}(Y)(q_Y) \times \text{sig}(\mathcal{A})(q_A) = \text{sig}(Y)(q_Y) \times \text{sig}(\tilde{A}^w)(\tilde{q}_{\tilde{A}^w} = \text{ren}_{sw}(q_A))$.

We note $C_X = (X, S_X) = \text{config}(X)(q_X), C_Y = (Y, S_Y) = \text{config}(Y)(q_Y), C_{\tilde{A}^w} = (\tilde{A}_{\tilde{A}^w}, S_{\tilde{A}^w}) = \text{config}(\tilde{A}^w)(q_{\tilde{A}^w})$. By construction of $\mu_Z, C_X = C_Y \cup C_{\tilde{A}^w}$ with $C_Y$ and $C_{\tilde{A}^w}$ compatible configuration (1).

We note $\varphi_X = \text{created}(X)(q_X)(a), \varphi_Y = \varphi_X \setminus \{A\}, \varphi_{\tilde{A}^w} = \emptyset, \varphi_Z = \varphi_X \cup \varphi_{\tilde{A}^w}$. If $a$ is $\mathcal{A}$-exclusive in state $q_X$, then $\varphi_X = \varphi_Y = \emptyset$.

- If $\mathcal{A}$ is not created by $a$, then $\varphi_X = \varphi_Z$.
- If $\mathcal{A}$ is created by $a$, then $\varphi_X = \varphi_Z \cup \{A\}$ and $\varphi_Z = \varphi_X \setminus \{A\}$

Since $X$ is a PCA and $(q_X, a, \eta(X,q_X,a)) \in D_X$, the constraint 2 of top/down transition preservation says that it exists a unique reduced configuration distribution $\eta'_X \text{ s. t. } \eta(X,q_X,a) \xrightarrow{f_X} \eta'_X$ with $f_X = \text{config}(X)$ and $\text{config}(X)(q_X) \Rightarrow \varphi_X \eta'_X$ (2).
For $Y$ (resp. $\tilde{A}^{sw}$) we note $\eta_Y = \eta(Y,q_Y,a)$ if $a \in \text{supp}(Y)(q_Y)$ and $\eta_Y = \delta_{q_Y}$ otherwise (resp. $\eta_{\tilde{A}^{sw}} = \eta(\tilde{A}^{sw},q_{\tilde{A}^{sw}},a)$ if $a \in \text{supp}(\tilde{A}^{sw})(q_{\tilde{A}^{sw}})$ and $\eta_{\tilde{A}^{sw}} = \delta_{q_{\tilde{A}^{sw}}}$ otherwise).

Since $Y$ and $\tilde{A}^{sw}$ are PCA, either because of the constraint 2 of top/down transition preservation or because $a$ is not action of the signature, it exists a unique reduced configuration distribution $\eta'_Y$ s. t. $\eta_Y \overset{f_Y}{\rightarrow} \eta'_Y$ with $f_Y = \text{config}(Y)$ and $\text{config}(Y)(q_Y) \implies \eta'_Y$ (resp. $\eta'_{\tilde{A}^{sw}}$ with $f_{\tilde{A}^{sw}} = \text{config}(\tilde{A}^{sw})$ and $\text{config}(\tilde{A}^{sw})(q_{\tilde{A}^{sw}}) \implies \eta'_{\tilde{A}^{sw}}$).

By construction $\forall(q'_Y,q'_{\tilde{A}^{sw}}) \in \text{states}(Y) \times \text{states}(\tilde{A}^{sw})$, $\text{constitution}(Y)(q'_Y) \cap \text{constitution}(\tilde{A}^{sw})(q'_{\tilde{A}^{sw}}) = \emptyset$ (and so $\text{auto}(\text{config}(Y)(q'_Y)) \cap \text{auto}(\text{config}(\tilde{A}^{sw})(q'_{\tilde{A}^{sw}})) = \emptyset$) which means $(***)$ base($C_A, a, \eta_Y$) $\cap$ base($\tilde{A}^{sw},a, \eta_{\tilde{A}^{sw}}$) $= \emptyset$.

The conjunction of (1), (2), (3) and (***)) allows us to apply the lemma 5.26. This means

- by item 4 of lemma 5.26: $\text{merge}((\eta'_A,\eta'_Y)) \overset{f_Z}{\rightarrow} \text{join}((\eta'_A,\eta'_Y))$ with $f_Z : C'_Y \mapsto (C'_Y, \tilde{C}'_{\tilde{A}^{sw}}, a) \ s. t. i) C'_Z \subseteq C'_Y \cup C'_A \cup \tilde{C}'_{\tilde{A}^{sw}}$, ii) $A \notin C'_Y$ and iii) $\forall B \neq A, B \notin \tilde{C}'_{\tilde{A}^{sw}}$ (4)

- by item 5 of lemma 5.26: $C_X \overset{a}{\rightarrow} \text{merge}((\eta'_A,\eta'_Y))$ (5)

Furthermore $\eta_{\tilde{A}^{sw}} = \eta_Y \otimes \eta_{\tilde{A}^{sw}}$. So by (3), $\eta_{\tilde{A}^{sw}} \overset{f_Z}{\rightarrow} \text{join}((\eta'_A,\eta'_Y))$ (***) with $f_Z : q'_Z = (q'_Y,q'_{\tilde{A}^{sw}}) \mapsto (\text{config}(Y)(q'_{A^{sw}}),\text{config}(\tilde{A}^{sw})(q'_{\tilde{A}^{sw}}))$.

Now we deal have to separate the treatment of the two cases:

- If $\tilde{A}$ is not created by $a$, since $\varphi_Z = \varphi_X$, because of (5) and (2), $\text{merge}((\eta'_A,\eta'_Y)) = \eta'_X$ and because of (2) $\eta(X,q_{X},a) \overset{f^{\text{X}}}{\rightarrow} \text{merge}((\eta'_A,\eta'_Y))$ (6). Because of (6) and (4), $\eta(X,q_{X},a) \overset{g}{\rightarrow} \text{join}((\eta'_A,\eta'_Y))$ with $g = f^{\text{Z}} \circ f^{\text{X}}$.

Hence, if $\tilde{A}$ is not created by $a \eta(X,q_{X},a) \overset{h}{\rightarrow} \eta(Z,q_{Z},a)$ with $h = (f^{\text{Z}})^{-1} \circ f^{\text{Z}} \circ f^{\text{X}} = \mu_{\tilde{Z}}$ which ends the proof for this case.

- If $\tilde{A}$ is created by $a$, we have both

  $C_X \overset{a}{ightarrow} \varphi_{X} \cup \{A\} \eta'_X$

  which means $C_X \overset{\nu}{\rightarrow} \eta'_X$ with

  - $\text{merge}((\eta'_A,\eta'_Y))$ generated by $\eta'_X$ and $\varphi_Z$ and
  - $\eta'_X$ generated by $\eta'_X$ and $\varphi_{X} \cup \{A\}$.

Thus $\eta'_X \overset{g^{\phi}}{\rightarrow} \text{merge}((\eta'_A,\eta'_Y))$ with $g^{\phi} : C'_X \subseteq C'_Y \cup \tilde{C}'_{\tilde{A}^{sw}} \mapsto \tilde{C}'_{\tilde{A}^{sw}}$, where $\tilde{C}'_{\tilde{A}^{sw}}(\{A\},S'_{\tilde{A}^{sw}} : A \mapsto \tilde{q}_{\tilde{A}^{sw}})$.

To summarize, we have:

- $\eta(X,q_{X},a) \overset{f^{\text{X}}}{\rightarrow} \eta'_X$

- $\eta'_X \overset{g^{\phi}}{\rightarrow} \text{merge}((\eta'_A,\eta'_Y))$

- $\text{merge}((\eta'_A,\eta'_Y)) \overset{f^{\text{Z}}}{\rightarrow} \text{join}((\eta'_A,\eta'_Y))$

- $\eta(Z,q_{Z},a) \overset{f^{\text{Z}}}{\rightarrow} \text{join}((\eta'_A,\eta'_Y))$

Hence $\eta(X,q_{X},a) \overset{h}{\rightarrow} \eta(Z,q_{Z},a)$ with $h^{\phi} = (f^{\text{Z}})^{-1} \circ f^{\text{Z}} \circ g^{\phi} \circ f^{\text{X}}$, i. e.

$f^{\phi} : q'_{X} \in \text{supp}(\eta(X,q_{X},a)) \mapsto (X,\varphi_{X})(q'_{X}), q'_{\tilde{A}^{sw}})$, which ends the proof for this case.
The second case where $\mathcal{A}$ is created will not be used before section 11.

We take advantage of the lemma 9.11 used for theorem 9.13 to introduce the notion of twin PCA and extends directly the lemma 9.11 and theorem 9.13 to twin PCA.

**Definition 9.9** ($X_\mathcal{A} \rightarrow X_\mathcal{A}'$). Let $X = (Q_X, \bar{q}_X, \text{sig}(X), D_X)$ be a PSIOA and $\bar{q}_X' \in \text{reachable}(X)$. We note $X_\mathcal{A} \rightarrow X_\mathcal{A}'$ the PSIOA $X' = (Q_X, \bar{q}_X', \text{sig}(X), D_X)$.

Two PCA $X$ and $X'$ are $\mathcal{A}$-twin if they differ only by their start state where one of them corresponds to $\mathcal{A}$-creation.

**Definition 9.10** ($\mathcal{A}$-twin). Let $\mathcal{A} \in \text{Autids}$. Let $X, X'$ be PCA. We say that $X' = X_\mathcal{A} \rightarrow X_\mathcal{A}'$, is a $\mathcal{A}$-twin of $X$ if it differs from $X$ at most only by its start states $\bar{q}_X$, reachable by $X$ s. t. either $X' = X$ or $\mathcal{A} \in \text{config}(X')(\bar{q}_X)$ and map(config($X'$)($\bar{q}_X$))($\mathcal{A}$) = $\bar{q}_\mathcal{A}$. If $X'$ is a $\mathcal{A}$-twin of $X$ and $Y = X \setminus \{\mathcal{A}\}$ and $Y' = X' \setminus \{\mathcal{A}\}$, we slightly abuse the notation and say that $Y'$ is a $\mathcal{A}$-twin of $Y$.

**Lemma 9.11** (partial surjectivity 1). Let $\mathcal{A} \in \text{Autids}$. Let $X$ be a PCA $\mathcal{A}$-conservative and $X'$ a $\mathcal{A}$-twin of $X$. Let $Y = X' \setminus \{\mathcal{A}\}$. Let $Y' = X' \setminus \{\mathcal{A}\}$. Let $\mathcal{A} \in \text{config}(X)(\bar{q}_X)$, (necessarily $X = X'$). Let $\mathcal{A} \notin \text{config}(X')(\bar{q}_X')$. Hence $\mu_{\mathcal{A}}(\bar{q}_X') = (\bar{q}_Y, \text{ren}_{\mathcal{A}}(\bar{q}_X'))$.

**Proof**. By induction on each prefix $\alpha^s = q_0^s, a_1, ..., a_k^s$ with $s \leq k$.

Base case: $s = 1$. Hence $\mu_{\mathcal{A}}(\bar{q}_X') = (\bar{q}_Y, \text{ren}_{\mathcal{A}}(\bar{q}_X'))$.

Induction: we assume this is true for $s$ and we show it implies this true for $s + 1$. We note $\bar{a}_s$ s. t. $\mu_{\mathcal{A}}(\bar{a}_s) = \alpha^s$. We also note $\bar{a}_s = \text{Istate}(\alpha^s)$ and we have by induction assumption $\mu_{\mathcal{A}}(\bar{a}_s) = q^s = (\bar{q}_Y^s, \bar{q}_X^s)$. Because of preservation of signature compatibility, $\text{sig}(X)(\bar{q}_X^s) = \text{sig}(Y)(\bar{q}_Y^s)$, $\text{sig}(\text{ren}_{\mathcal{A}}(\bar{q}_X^s)) = (\bar{q}_Y^s, \text{ren}_{\mathcal{A}}(\bar{q}_X^s))$. Hence $\alpha^{s+1} \in \text{sig}(X)(\bar{q}_X^s)$. From the previous lemma it exists $\bar{a}_{s+1} \in \text{Istate}(\bar{a}_s)$ and hence $\alpha^{s+1} \in \text{Istate}(\alpha^s)$. Finally, since no creation of and from $\mathcal{A}$ occurs by assumption of presence of $\mathcal{A}$, we can use lemma 9.8 of homomorphic transition which gives $\eta_{(X, \bar{q}^s, \alpha^{s+1})} \Leftrightarrow \eta_{(Z, q^s, \alpha^{s+1})}$ which means $\bar{a}^{s+1} \in \text{Frag}(\bar{a}_s)$ which ends the induction and so the proof.

Before using lemma 9.11 and 9.7 to demonstrate theorem 9.13 of partial compatibility after reconstruction, we take the opportunity to extend lemma 9.11:

**Lemma 9.12** (partial surjectivity 2). Let $\mathcal{A} \in \text{Autids}$. Let $X$ be a PCA $\mathcal{A}$-conservative. Let $Y = X \setminus \mathcal{A}$. Let $Y'$ be a $\mathcal{A}$-twin of $Y$. Let $Z = Y' \setminus \mathcal{A}$.

Let $\alpha = q_0^s, a_1, ..., q_k^s$ be an action of $Z$. Let assume (a) $q_{\mathcal{A}}^s \in \text{ren}_{\mathcal{A}}(\bar{q}_X)$. Then it exists $\bar{a} \in \text{Frag}(X)$, s. t. $\mu_{\mathcal{A}}(\bar{a}) = \alpha$. If $Y' = Y$, it exists $\bar{a} \in \text{Exec}(X)$, s. t. $\mu_{\mathcal{A}}(\bar{a}) = \alpha$.

**Proof**. We already know this is true up to $k^*$ because of lemma 9.11. We perform the same induction than the one of the previous lemma on partial surjectivity: We note $\bar{a}_s$ s. t. $\mu_{\mathcal{A}}(\bar{a}_s) = \alpha^s$. We also note $\bar{q}^s = \text{Istate}(\alpha^s)$ and we have by induction assumption $\mu_{\mathcal{A}}(\bar{a}_s) = q^s = (\bar{q}_Y^s, \bar{q}_X^s)$. Because of preservation of signature compatibility, $\text{sig}(X)(\bar{q}_X^s) = \text{sig}(Y)(\bar{q}_Y^s) \times \text{sig}(\text{ren}_{\mathcal{A}}(\bar{q}_X^s))$. Hence $\alpha^{s+1} \in \text{sig}(X)(\bar{q}_X^s)$. Now we use the assumption (c), that says that $\mathcal{A} \notin \text{created}(X)(\bar{q}_X^s)(\alpha^{s+1})$ to be able to apply preservation of transition since no creation of $\mathcal{A}$ can occurs.
Now we can use lemma 9.11 and 9.7 to demonstrate theorem 9.13 of partial compatibility after reconstruction.

**Theorem 9.13 (Partial-compatibility after reconstruction).** Let $A \in \text{Autids}$. Let $X$ be a PCA $A$-conservative s. t. $\forall q_X \in \text{states}(X)$, for every action a $A$-exclusive in $q_X$, created($X$)($q_X$)($a$) = $\emptyset$. Let $X'$ be a $A$-twin of $X$ and $Y' = X' \setminus \{A\}$. Then $Y'$ and $\tilde{A}^{tw}$ are partially-compatible.

**Proof.** Let $Z' = (Y', \tilde{A}^{tw})$. Let $\alpha$ be a pseudo-execution of $Z'$ with $\text{Lstate}(\alpha) = q_Z = (q_{Y'}, q_{\tilde{A}^{tw}})$. Case 1) $q_{\tilde{A}^{tw}} = q_\phi$. The compatibility is immediate since $\text{sig}(\tilde{A}^{tw})(q_\phi) = \emptyset$. Case 2) $q_{\tilde{A}^{tw}} \neq q_\phi$. Since $\mu(X)$ cannot be re-created after destruction by neither $Y$ or $\tilde{A}^{tw}$ and (**) $\forall q_X \in \text{states}(X)$, for every action a $A$-exclusive in $q_X$, created($X$)($q_X$)($a$) = $\emptyset$ we can use the previous lemma 9.11 to show it exists $\tilde{a} \in \text{Execs}(X')$, s. t. $\mu_{\tilde{a}}(\tilde{a}) = a$. Thus, $\text{Lstate}(\alpha) = \mu_{\tilde{a}}(\text{Lstate}(\tilde{a}))$ which means $Z'$ is partially-compatible at $\text{Lstate}(\alpha)$ by lemma 9.7. Hence $Z$ is partially-compatible at every reachable state, which means $Y'$ and $\tilde{A}^{tw}$ are partially-compatible. We can legitimately note $Z' = Y'||\tilde{A}^{tw}$.

Since $Z' = (Y', \tilde{A}^{tw})$ is partially-compatible, we can legitimately note $Z' = Y'||\tilde{A}^{tw}$, which will be the standard notation in the remaining.

### 9.3 Execution-matching from $X$ to $X \setminus \{A\}||\tilde{A}^{tw}$

In this subsection, we show in theorem 9.19 that $\mu_{\tilde{A}}^X$ is a (incomplete) PCA executions-matching from $X$ to $(X \setminus \{A\})||\tilde{A}^{tw}$ in a restricted set of executions of $X$ that do not create $A$.

We start by defining the restricted set of executions of $X$ that do not create $A$ with definitions 9.14 and 9.15.

**Definition 9.14 (execution without creation).** Let $A$ be a PSIOA. Let $X$ be a PCA, we write $\text{execs-}\text{without-}\text{creation}(X)(A)$ the set of executions of $X$ without creation of $A$, i. e. $\text{execs-}\text{without-}\text{creation}(X)(A) = \{\alpha = q_0a_1\ldots a_{k-1}a_k \in \text{Execs}(X) | \forall i \in [0, |\alpha|], A \notin \text{uts(config}(X)(q^i))\}$.

**Definition 9.15 (reachable-by).** Let $X$ be a PSIOA or a PCA. Let $\text{Execs}'(X) \subseteq \text{Execs}(X)$. We note $\text{reachable-by}(\text{Execs}'(X))$ the set of states of $X$ reachable by an execution of $\text{Execs}'(X)$, i. e. $\text{reachable-by}(\text{Execs}'(X)) = \{q \in \text{states}(X) | \exists \alpha \in \text{Execs}'(X), \text{Lstate}(\alpha) = q\}$.

The next 2 lemma show that reconstruction preserves configuration and signature. They will be sufficient to show that the restriction of $\mu_{\tilde{A}}^X$ on $\text{reachable-by}(\text{execs-}\text{without-}\text{creation}(X)(A))$ is a PCA executions-matching.

**Lemma 9.16 ($\mu_{\tilde{A}}$ configuration preservation).** Let $A \in \text{Autids}$. Let $X$ be a $A$-conservative PCA, $Y = X \setminus A$, $Z = Y||\tilde{A}^{tw}$. Let $q_X \in \text{states}(X)$, $q_Z = (q_Y, q_{\tilde{A}^{tw}}) \in \text{states}(Z)$ s. t. $\mu_{\tilde{A}}(q_X) = q_Z$. Then $\text{config}(X)(q_X) = \text{config}(Z)(q_Z)$.

**Proof.** By definition of composition of PCA, $\text{config}(Z)(q_Z) = \text{config}(Y)(q_Y) \cup \text{config}(\tilde{A}^{tw})(q_{\tilde{A}^{tw}})$. (*) Also, by $\mu_{\tilde{A}}$-correspondence, $\text{config}(X)(q_X) \setminus A = \text{config}(Y)(q_Y)$ (**).

We deal with the two cases $\text{sig(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) = 0}$ or $\text{sig(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) \neq 0}$

- If $\text{sig(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) = 0}$, then $A \notin \text{uts(config}(X)(q_X))$ which means, that $\text{config}(X)(q_X) = \text{config}(X)(q_X) \setminus A$ (1). Furthermore, $\text{config}(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) = (0, 0)$ (2). Because of (***) and (1), $\text{config}(Y)(q_Y)$ and because of (**) and (2), $\text{config}(X)(q_X) = \text{config}(Z)(q_Z)$.

- If $\text{sig(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) \neq 0}$, then $A \in \text{uts(config}(X)(q_X))$. We note $C_A = \text{config}(\tilde{A}^{tw})(q_{\tilde{A}^{tw}}) = (\{A\}, S : A \mapsto \text{map(config}(X)(q_X))(A))$. By (**), $\text{config}(Z)(q_Z) = \text{config}(Y)(q_Y) \cup C_A$ and by (**) $\text{config}(Y)(q_Y) \cup C_A = \text{config}(X)(q_X) \setminus A \cup C_A = \text{config}(X)(q_X)$. Hence, $\text{config}(X)(q_X) = \text{config}(Z)(q_Z)$. 

72
Thus in all cases, \( \text{config}(X)(q_X) = \text{config}(Z)(q_Z) \) which ends the proof.

\[
\text{LEMMA 9.17 (\( \mu_\pi \) signature-preservation). Let } A \in \text{Autids}. \text{ Let } X = X \setminus \{A\}, Z = Y || A^\pi w. \text{ Let } q_X \in \text{states}(X), q_Z = (q_Y, q_{A^\pi w}) \in \text{states}(Z) \text{ s.t. } \mu_\pi(q_X) = q_Z. \text{ Then } \text{sig}(X)(q_X) = \text{sig}(Z)(q_Z).
\]

\[
\text{PROOF. By lemma 9.6 of preservation of signature } \text{sig}(X)(q_X) = \text{sig}(Y)(q_Y) \times \text{sig}(A^\pi w)(q_{A^\pi w}). \text{ By definition of composition of PCA, } \text{sig}(Z)(q_Z) = \text{sig}(Y)(q_Y) \times \text{sig}(A^\pi w)(q_{A^\pi w}) \text{ which ends the proof.}
\]

Now we can state our strong PCA executions-matching:

\[
\text{Definition 9.18. Let } A \text{ be a PSIOA. Let } X = X \setminus \{A\} \text{ and } Z = Y || A^\pi w. \text{ We define } (X,\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, X,\tilde{\mu}_e^A) \text{ (notated } (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \text{ when it is clear in the context) as follows:
\]

- \( \tilde{\mu}_A^A \) the restriction of \( \mu_\pi^A \) on reachable-by(execs-without-creation)(X)(A).
- \( f^{\pi'}: (q_X, a, \eta(q_X, a)) \in D'_X \mapsto (\tilde{\mu}_A^A(q_X), a, \eta(Z,\tilde{\mu}_A^A(q_X), a)) \) where \( D'_X = \{(q_X, a, \eta(q_X, a)) \in D_X | q_X \in \text{reachable-by(execs-without-creation)(X)(A))} \).
- \( \tilde{\mu}_e^A \) the restriction of \( \mu_e^A \) on execs-without-creation(X)(A).

\[
\text{Theorem 9.19 (execution-matching after reconstruction). Let } A \text{ be a PSIOA. Let } X = X \setminus \{A\}. \text{ The triplet } (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \text{ is a strong PCA executions-matching from } X \text{ to } Y || A^\pi w \text{ if } A \in \text{auts(config}(X,\tilde{\mu}_A^A)(\text{start}(X,\tilde{\mu}_A^A))) \text{ and from } X \text{ to } Y || A^\pi w \text{ otherwise.}
\]

\[
\text{PROOF. We note } Z = Y || A^\pi w \text{ and } Z^\phi = Y || A^\pi w \text{ otherwise.}
\]

- \( \tilde{\mu}_A^A \) is a strong PCA-state-matching since
  - starting state preservation is ensured by construction:
    - \( A \in \text{auts(config}(X,\tilde{\mu}_A^A)(\text{start}(X,\tilde{\mu}_A^A))) \): \( \tilde{\mu}_A^A(q_X) = \tilde{q}_Z \)
    - \( A \notin \text{auts(config}(X,\tilde{\mu}_A^A)(\text{start}(X,\tilde{\mu}_A^A))) \): \( \tilde{\mu}_A^A(q_X) = \tilde{q}_Z^\phi \)
  - signature preservation is ensured \( \forall (q_X, q_Z) \in \text{states}(X) \times \text{states}(Z) \text{ s.t. } q_Z = \tilde{\mu}_A^A(q_X), \text{sig}(X)(q_X) = \text{sig}(Z)(q_Z) \) by lemma 9.17 of signature preservation of \( \mu_\pi \).
- \( D'_X = \text{dom}(\tilde{\mu}_A^A) \) is eligible to PCA transition-matching (and thus \( (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \) is a strong PCA-transition-matching) since
  - matched state preservation is ensured: \( \forall \eta(q_X, q_X, a) \in D'_X, q_X \in \text{dom}(\tilde{\mu}_A^A) \text{ by construction of } D'_X \)
  - equitable corresponding distribution is ensured: \( \forall \eta(q_X, q_X, a) \in D'_X, q_X \in \text{supp}(\eta(q_X, q_X, a)) \text{ if } q'' = \eta(Z,\tilde{\mu}_A^A(q_X), a) \text{ then } \tilde{\mu}_e^A(q''') = \tilde{\mu}_e^A(q'') \) by lemma 9.8 of homomorphic transition.
- \( (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \) is the PCA-execution-matching induced by \( (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A) \) and correctly verifies:
  - For each state \( q \) in an execution in execs-without-creation(X)(A), \( q \in \text{dom}(\tilde{\mu}_A^A) \).

Then, the triplet \( (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \) is a strong PCA-executions-matching from \( X \) to \( Z \) if \( A \in \text{auts(config}(X,\tilde{\mu}_A^A)(\text{start}(X,\tilde{\mu}_A^A))) \) \( \tilde{\mu}_A^A(q_X) = \tilde{q}_Z \) and from \( X \) to \( Z^\phi \) otherwise.

\[
\text{extension and continuation of } (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A). \text{ Now, we continue the executions-matching } (\tilde{\mu}_A^A, \tilde{\mu}_\pi^A, \tilde{\mu}_e^A) \text{ to deal with } A \text{ creation at very last action.}
\]
Definition 9.20 (Preparing continuation of PCA executions-matching from X to Z). Let $\mathcal{A}$ be a PSIOA. Let $X$ be a $\mathcal{A}$-conservative PCA. We define

- $\tilde{\mu}_{\mathcal{A}}^{x+} : q_X \in \text{reachable-by}(\text{execs-with-only-one-creation-at-last-action}(X)(\mathcal{A})) \mapsto (\tilde{\mu}_{\mathcal{A}}(q_{X, \alpha}), q_{X, \alpha})$.
- $\bar{\mu}_{tr}^{x+} : (q_X, a, \eta_{(X, q_{X, a})}) \in \text{dom}(\tilde{\mu}_{tr}^{x+}) \cup D' \mapsto (\bar{\mu}_{tr}(q_X, a, \eta_{(X, \tilde{\mu}_{\mathcal{A}}(q_{X, a})), a})$ where $D' = \{(q_X, a, \eta_{(X, q_{X, a})}) \in \text{trans}(X)[q_X] \in \text{reachable-by}(\text{execs-with-out-creation-at-last-action}(X)(\mathcal{A})) \land \mathcal{A} \not\in \text{autos}(\text{config}(X)(q_X)) \land \mathcal{A} \in \text{created}(X)(q_X)(a)\}$

We show that $\text{dom}(\tilde{\mu}_{tr}^{x+}) \setminus \text{dom}(\bar{\mu}_{tr}^{x+})$ verifies the equitable corresponding property of definition 7.7.

Lemma 9.21 (Continuation of PCA transitions-matching from X to Z). Let $\mathcal{A}$ be a PSIOA. Let $X$ be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y \| \tilde{\mathcal{A}}^w$.

$\forall (q_{X, a}, \eta_{(X, q_{X, a})}) \in \text{dom}(\tilde{\mu}_{tr}^{x+}) \setminus \text{dom}(\bar{\mu}_{tr}^{x+})$, $\forall q'_X \in \text{supp}(\eta_{(X, q_{X, a})})$, $\eta_{(X, q_{X, a})}(q'_X) = \eta_{(Z, \tilde{\mu}_{\mathcal{A}}(q_{X, a}))}(q'_X)$

Proof. By configuration preservation, $\text{Conf} = \text{config}(X)(q_X) = \text{config}(Z)(\tilde{\mu}_{\mathcal{A}}(q_X))$. We have $\text{Conf} \rightarrow_{\mathcal{A}} \eta_{(\text{Conf}, a)}$. Moreover, by $\mu$-correspondence rule, $\phi_X \setminus \{\mathcal{A}\} = \phi_Z$, with $\phi_X = \text{created}(X)(q_X)(a)$ and $\phi_Z = \text{created}(Z)(\tilde{\mu}_{tr}^{x+}(q_X))(a)$.

Hence $\text{Conf} \rightarrow_{\mathcal{A}} \eta_X$' with $\eta_X$ generated by $\phi_X$ and $\eta_{(\text{Conf}, a)}$, while $\text{Conf} \rightarrow_{\mathcal{A}} \eta'_Z$ with $\eta'_Z$ generated by $\phi_Z$ and $\eta_{(\text{Conf}, a)}$.

Since $\mathcal{A}$ is created, for every $\text{Conf}_Z'' = (A'_Z, S'_Z)$ with $\mathcal{A} \notin A'_Z$, for every $\text{Conf}_X'' = (A'_X, S'_X)$ with $A'_X = A' \cup \{\mathcal{A}\}$ where $S'_X(\mathcal{A}) = \tilde{\mathcal{A}}$ and $S'_X$ agrees with $S'_Z$ on $A'_Z$, $\eta'_Z(\text{Conf}_Z''(\mathcal{A})) = \eta'_X(\text{Conf}_X''(\mathcal{A}))$, while $\eta'_X(\text{Conf}_X''(\mathcal{A})) = 0$ for every $\text{Conf}_X'' = (A'_X, S'_X)$ s.t either $\mathcal{A} \notin A'_X$ or $\mathcal{A} \in A'_X$ but $S'_X(\mathcal{A}) \neq \tilde{\mathcal{A}}$. So $\eta_{Z, \tilde{\mu}_{\mathcal{A}}(q_{X, a})}(\bar{\mu}_{tr}^{x+}(q'_X)) = \eta_{Z, \text{config}(Z)(\tilde{\mu}_{tr}^{x+}(q'_X))) = \eta_X'(\text{config}(X)(q'_X))) = \eta_{X, q_{X, a}}(q'_X)$ which ends the proof.

Since $\text{dom}(\tilde{\mu}_{tr}^{x+}) \setminus \text{dom}(\bar{\mu}_{tr}^{x+})$ verifies the equitable corresponding property of definition 7.7, we can define a continuation of $(\tilde{\mu}_{tr}^{x+}, \bar{\mu}_{tr}^{x+}, \mu_e^{x+})$ that deal with $\mathcal{A}$-creation at very last action.

Definition 9.22 (Continuation of PCA executions-matching from X to Z). Let $\mathcal{A}$ be a PSIOA. Let $X$ be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y \| \tilde{\mathcal{A}}^w$. Let $\mathcal{D}' = \text{dom}(\tilde{\mu}_{tr}^{x+}) \setminus \text{dom}(\bar{\mu}_{tr}^{x+})$. Since $\forall (q_X, a, \eta_{(X, q_{X, a})}) \in \mathcal{D}'$, $\forall q'_X \in \text{supp}(\eta_{(X, q_{X, a})})$, $\eta_{(X, q_{X, a})}(q'_X) = \eta_{Z, \tilde{\mu}_{\mathcal{A}}(q_{X, a})}(\bar{\mu}_{tr}^{x+}(q'_X))$ by previous lemma 9.21, we can define:

$((\tilde{\mu}_{tr}^{x+}, \bar{\mu}_{tr}^{x+}, \mu_e^{x+}))$ the $(\tilde{\mu}_{tr}^{x+}, D'_X)$-continuation of $(\tilde{\mu}_{tr}^{x+}, \bar{\mu}_{tr}^{x+}, \mu_e^{x+})$.

We terminate this subsection by showing the $\tilde{\mathcal{E}}$-extension of our continued PCA executions-matching is always well-defined.

Theorem 9.23 (Extension of continued executions-matching after reconstruction). Let $\mathcal{A}$ be a PSIOA. Let $X$ be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y \| \tilde{\mathcal{A}}^w$. Let $\tilde{\mathcal{E}}$ partially-compatible with both $X$ and $Z$. The $\tilde{\mathcal{E}}$-extension of $(X, \tilde{\mu}_{tr}^{x+}, X, \tilde{\mu}_{tr}^{x+}, X, \tilde{\mu}_{tr}^{x+}, \tilde{\mu}_{tr}^{x+}, \tilde{\mu}_{tr}^{x+})$, noted $(((\tilde{\mathcal{E}} \| X), \tilde{\mu}_{tr}^{x+}, (\tilde{\mathcal{E}} \| X), \tilde{\mu}_{tr}^{x+}, \tilde{\mu}_{tr}^{x+}, (\tilde{\mathcal{E}} \| X), \tilde{\mu}_{tr}^{x+})$, is a strong continued PCA executions-matching from $\tilde{\mathcal{E}} \| X$ to $\tilde{\mathcal{E}} \| Z$.

Proof. By definition of $\tilde{\mu}_{tr}^{x+}$ and $\mu_{tr}^{x+}$, we have

- $\tilde{\mathcal{E}} \| X = \text{execs-without-creation}(\tilde{\mathcal{E}} \| X)(\mathcal{A})$
- $\tilde{\mathcal{E}}_X = \text{execs-with-only-one-creation-at-last-action}(\tilde{\mathcal{E}} \| X)(\mathcal{A})$
This section aims to show in theorem 9.24 that operation of projection/deprivation and composition are commutative.

\[ Q_\text{\textbf{E}}(X) = \text{reachable-by}(\tilde{E}_\text{\textbf{E}})(X) \]

\[ Q_\text{\textbf{E}}(X) = \text{reachable-by}(\tilde{E}_\text{\textbf{E}}^*_X) \]

\[ Q_\text{\textbf{X}} = \text{reachable-by}(\tilde{E}_\text{\textbf{X}}) \]

\[ Q_\text{\textbf{X}}^* = \text{reachable-by}(\tilde{E}_\text{\textbf{X}}^*_X) \]

This allow us to apply lemma 7.17 of "sufficient conditions to obtain range inclusion" to both \((\tilde{E}_\text{\textbf{E}})(X), \tilde{\mu}_z^{\text{A,}+} \) and \((\tilde{E}_\text{\textbf{E}})(X), \tilde{\mu}_z^A \) which gives \( \text{range}((\tilde{E}_\text{\textbf{E}})(X), \tilde{\mu}_z^{\text{A,}+}) \subseteq \text{states}(\tilde{E}_\text{\textbf{E}})(Z) \) and \( \text{range}((\tilde{E}_\text{\textbf{E}})(X), \tilde{\mu}_z^A) \subseteq \text{states}(\tilde{E}_\text{\textbf{E}})(Z) \) which allows us to apply lemma 7.24.

The lemma 7.34 implies that the resulting executions-matching is a strong one.

\[ \Box \]

### 9.4 Composition and projection are commutative

This section aims to show in theorem 9.24 that operation of projection/deprivation and composition are commutative.

**Theorem 9.24 ((X||E) \ \{A\} and (X \ \{A\})||E are semantically equivalent).** Let \( A \) be a PSIOA. Let \( X \) be a \( A \)-fair PCA partially-parallel with \( E \) that never counts \( A \) in its constitution with both \( X, E \) and \( X||E \) configuration-conflict-free.

The PCA \((X||E) \ \{A\} \) and \((X \ \{A\})||E \) are semantically equivalent.

**Proof.** We note \( W = X||E, U = (X||E) \ \{A\}, V = (X \ \{A\})||E, \tilde{\mu}_s^{X,\text{A}} = X\tilde{\mu}_s^A, \mu_s^{W,\text{A}} = W\mu_s^A \). To stay simple, we note \( \text{Id} \) the identity function on any domain, that is we note \( \text{Id} \) for both \( \text{Id}_E : q_E \in \text{states}(E) \mapsto q_E \) and \( \text{Id}_U : q_U \in \text{states}(U) \mapsto q_U \).

The plan of the proof is the following one:

- We will construct two functions, \( \text{iso}_{UV} : \text{states}(U) \rightarrow \text{states}(V) \) and \( \text{iso}_{U} : \text{states}(V) \rightarrow \text{states}(U) \), s. t. \( \text{iso}_{UV}(q_U) \) is the unique element of \((\mu_s^{X,\text{A}}, \text{Id})(\mu_s^{W,\text{A}})^{-1}(q_U)\) and \( \text{iso}_{U}(q_V, q_E) \) is the unique element of \( \mu_s^{W,\text{A}}((\mu_s^{X,\text{A}}, \text{Id})^{-1}(q_V, q_E))\).

- Then we will show that \( \text{iso}_{UV} \) and \( \text{iso}_{U} \) are two bijections s. t. \( \text{iso}_{U} = \text{iso}_{UV}^{-1} \).

- Thereafter we will show that for every \((q_U, q_V), (q'_U, q'_V) \in \text{states}(U) \times \text{states}(V)\), s. t. \( q_U = \text{iso}_{UV}(q'_U) \) and \( q'_V = \text{iso}_{UV}(q'_U) \), then \( q_U \text{strict} q_V, q'_U \text{strict} q'_V \) and for every \( a \in \text{\tilde{\text{Supp}}}(U)(q_U) = \text{\tilde{\text{Supp}}}(V)(q_V), \eta(U, q_U, a)(q'_U) = \eta(V, q_V, a)(q'_V) \).

- Finally, it will allow us to construct a strong complete bijective execution-matching induced by \( \text{iso}_{UV} \) and \( D_U \) (the set of discrete transitions of \( U \)) in bijection with a strong complete bijective execution-matching induced by \( \text{iso}_{UV} \) and \( D_V \) (the set of discrete transitions of \( V \)).

First, we show that for every \( q_W = (q_X, q_E) \in \text{reachable}(W) \subset \text{states}(X) \times \text{states}(E)\), the state \( q_V = (\mu_s^{X,\text{A}}, \text{Id})(q_W) = (\mu_s^{X,\text{A}}(q_X), q_E) \) is an element of \( \text{reachable}(V) \) (*). We proceed by induction. Basis: \((\mu_s^{X,\text{A}}(q_X), q_E) \) is the initial state of \( V \). Induction: Let \( q_W \triangleq (q_X, q_E), q'_W \triangleq (q_X, q'_E) \in \text{reachable}(W), q_V \in \text{reachable}(V), a \in \text{\tilde{\text{Supp}}}(W)(q_W) \) s. t. \( q'_W \in \text{\tilde{\text{Supp}}}(V)(q_V, a) \), \( q_V = (\mu_s^{X,\text{A}}, \text{Id})(q_W) \), and \( q'_V = (\mu_s^{X,\text{A}}, \text{Id})(q'_W) \). There is two cases:
case 1) $a$ is $\mathcal{A}$-exclusive in $q_W$. In this case $q_W \setminus \{a\} q''_W$, which means $q''_W = q_V$ and ends the proof

case 2) $a \in \overline{\text{sg}}(V)(q_V) \cap \overline{\text{sg}}(W)(q_W)$

We need to show that $q''_V \in \text{supp}(\eta(W,q_{V,a}))$. This is easy to show. Indeed, $q''_V \in \text{supp}(\eta_W(\eta_{W,q_{V,a}}))$ means $(q'_X, q'_E) \in \text{supp}(\eta(X,q_{X,a}) \oplus \eta(E,q_{E,a}))$ (with the convention $\eta(X,q_{X,a}) = \delta_X$ if $q \not \in \overline{\text{sg}}(X)(q_X)$) and $\eta(E,q_{E,a}) = \delta_E$ if $a \not \in \overline{\text{sg}}(E)(q_E)$ which means $q'_X \in \text{supp}(\eta(X,q_{X,a}))$ and $q'_E \in \text{supp}(\eta(E,q_{E,a}))$. So $\mu^W_{\mathcal{A}}(q''_V) = \text{supp}(\eta(Y,\mu^W_{\mathcal{A}}(q_X,a)))$ which means $(\mu^W_{\mathcal{A}}(q'_X), q'_E) \in \text{supp}(\eta(Y,\mu^W_{\mathcal{A}}(q_X,a)))$, that is $(\mu^W_{\mathcal{A}}(q'_X), q'_E) \in \text{supp}(\eta(V,\mu^W_{\mathcal{A}}(q_X,q_{V,a}),\eta(E,q_{E,a}))$ and thus $q''_V \in \text{supp}(\eta(V,q_{V,a}))$ so $q''_V \in \text{reachable}(V)$.

Second, we show that for every $q_V \in (q_Y, q_E) \in \text{reachable}(V)$, it exists $q_W \in (q_X, q_E) \in \text{reachable}(W)$ s. t. $q_V = (\mu^W_{\mathcal{A}}, Id)(q_W)$. The reasoning is the same, we proceed by induction. The basis is performed with start state correspondence as before. Induction: Let $q_V \in (q_Y, q_E), q'_V \in (q'_Y, q'_E) \in \text{reachable}(V), q_W \in \text{reachable}(W), a \in \overline{\text{sg}}(V)(q_Y) \cap \overline{\text{sg}}(W)(q_W)$ s. t. $q''_V \in \text{supp}(\eta(V,q_{V,a}))$ with $q_V = (\mu^W_{\mathcal{A}}, Id)(q_W)$.

We need to show that exists $q''_W \in \text{supp}(\eta(W,q_{W,a}))$ s. t. $q''_V = (\mu^W_{\mathcal{A}}, Id)(q''_W)$. This is easy to show because of $\mu^W_{\mathcal{A}}$ -correspondence. For every $q''_V \in (q'_V, q_E) \in \text{supp}(\eta(V,q_{V,a},q_Y), q'_V \in \text{support}(\eta(V,q_{V,a},q_Y))$. Because of $\mu^W_{\mathcal{A}}$ -correspondence, it exists $q''_W \in \text{supp}(\eta(X,q_{X,a}))$ with $q''_W = (\mu^W_{\mathcal{A}}, q' X)$, thus it exists $q''_W = (q' X, q'_E) \in \text{supp}(\eta(W,q_{W,a}))$ s. t. $q''_W = (\mu^W_{\mathcal{A}}, q' X, q'_E)$ which ends the proof of this second point.

Now we can construct $\text{iso}_{U,W}$ and $\text{iso}_{V,U}$.

- $\text{iso}_{U,W}$: for every $q_U \in \text{states}(U)$, $(\mu^U_{\mathcal{A}}, Id)^{-1}(q_U) \neq \emptyset$ by construction of $U$,
and for every $q_W \in (q_X, q_E), q''_W \in (q'_X, q'_E) \in (\mu^W_{\mathcal{A}}, q''_W) \in (\mu^W_{\mathcal{A}}, q''_W)[\mathcal{A}]^{-1}(q_U)$, $q''_W \in \text{reachable}(q_W)$

- $\text{iso}_{V,U}$: for every $q_V \in (q_Y, q_E) \in \text{states}(V)$, $(\mu^V_{\mathcal{A}}, Id)^{-1}(q_V) \neq \emptyset$ by (*). For every $q_V \in (q_Y, q_E), q''_V \in (q'_Y, q'_E) \in (\mu^V_{\mathcal{A}}, q''_V) \in (\mu^V_{\mathcal{A}}, q''_V)[\mathcal{A}]^{-1}(q_V)$, $q''_V \in \text{reachable}(q_V)$, which means $q''_V \in \text{reachable}(q_V)$ and so $q''_V \in (\mu^V_{\mathcal{A}}, q''_V) [\mathcal{A}]^{-1}(q_V) = (q_V)$ where $q''_V \in \text{iso}_{U,W}(q_V) \in \text{states}(U)$

Now we can show that $\text{iso}_{U,W}$ is a bijection with $\text{iso}_{V,U}$.

- surjectivity of $\text{iso}_{V,U}$: Let $q_V \in (q_Y, q_E) \in \text{reachable}(V)$, we will show that it exists $q_U \in \text{reachable}(U)$ s. t. $\text{iso}_{V,U}(q_V) = q_U$. Indeed, we already know that (*) it exists $q_W = (q_X, q_E) \in (\mu^W_{\mathcal{A}}, Id)^{-1}(q_V)$ \text{reachable}(W).

Let $q_U = (\mu^U_{\mathcal{A}}, q''_U)$. By construction of $U$, we have $q_U \in \text{reachable}(U)$ and $q_W \in (\mu^W_{\mathcal{A}}, q''_W)$ and $(\mu^U_{\mathcal{A}}, Id)(q_W) = q_V$ which means $\text{iso}_{U,W}(q_U) = q_V$ and ends this item.

- injectivity of $\text{iso}_{U,W}$: Let $q_V \in \text{reachable}(V)$, $q''_V \in \text{reachable}(U)$ s. t. $\text{iso}_{U,W}(q_V) = \text{iso}_{V,U}(q''_V)$ then $q_U = q''_U$. Again for every $q_V, q'_V \in (\mu^W_{\mathcal{A}}, Id)^{-1}(q_V), q''_V \in \text{reachable}(q''_V)$, which means $(\mu^W_{\mathcal{A}}, q''_V) = (\mu^W_{\mathcal{A}}, q''_V)$.

But for every $q_V, q''_V \in \text{reachable}(q_V), q''_V \in \text{reachable}(q''_V)$, which means $q''_V = q''_U$.

Let (i) $q_V \in \text{iso}_{U,W}(q_U)$ or (ii) $q''_V = \text{iso}_{V,U}(q''_V)$ we will show that in both (i) and (ii) $q''_V \in \text{reachable}(q''_V)$ By definition, $\{q_V\} = (\mu^U_{\mathcal{A}}, Id)(\mu^W_{\mathcal{A}}, q''_V$)

In case (ii) we note $q_{W,V}$ an arbitrary element of $(\mu^W_{\mathcal{A}}, q''_V) \neq \emptyset$, while in case (ii) we note $q_W$ an arbitrary element of $(\mu^W_{\mathcal{A}}, q''_V) \neq \emptyset$. In both cases, we have 1a) $\text{config}(W)(q_W) \setminus \{\mathcal{A}\} = \text{config}(U)(q_V)$ and 1b) $\text{config}(W)(q_W) \setminus \{\mathcal{A}\} = \text{config}(V)(q_V)$, which means 1c) $\text{config}(U)(q_V) = \text{config}(V)(q_V)$. Then we have 2a) hidden-actions($W(q_W) \setminus \text{pot-out}(W)(q_W) \setminus \mathcal{A}) = \text{hidden-actions}(U)(q_V) \setminus \text{pot-out}(W)(q_W) \setminus \mathcal{A}) = \text{hidden-actions}(U)(q_V)$ and 2b) hidden-actions($W(q_W) \setminus \text{pot-out}(W)(q_W) \setminus \mathcal{A}) = \text{hidden-actions}(V)(q_V) \setminus \text{pot-out}(W)(q_W) \setminus \mathcal{A}) = \text{hidden-actions}(V)(q_V)$.
which means 2c) hidden-actions\((U)(q_U) = hidden-actions(V)(q_V)\). Thereafter we have 3a) for every action \(a \in sig(W)(q_W) \cap sig(U)(q_U)\), created\((W)(q_W)(a) | A = created\((U)(q_U)(a) | A\) = created\((U)(q_U)(a) | A\) and 3b) for every action \(a \in sig(W)(q_W) \cap sig(V)(q_V)\), created\((W)(q_W)(a) | A = created\((V)(q_V)(a) | A\) = created\((V)(q_V)(a) | A\) which means 3c) for every action \(a \in sig(U)(q_U) = sig(V)(q_V)\), created\((U)(q_U)(a) = created\((V)(q_V)(a)\). The conjunction of 3a), 3b) and 3c) lead us to \(qv R _{strict} qU\).

Now we can show that \(iso_{UV}\) is the reverse function of \(iso_{UV}: Let (q_U, q_V) \in reachable(U) \times reachable(V) s.t. q_V = iso_{UV}(q_U)\). We need to show that \(iso_{UV}(q_V) = q_U\). The point is that it exists a unique \(q'_U \triangleq iso_{UV}(q_V)\) and we have \(qv R _{strict} qU\) and \(qv R _{strict} q'_U\) which means \(qv R _{strict} q'_U\) and so \(q U = q'_U\) by assumption of configuration-conflict-free PCA. Hence \(iso_{UV} = iso_{VU}^{-1}\).

The last point is to show that for every \((q_U, q_V), (q'_U, q'_V) \in reachable(U) \times reachable(V), s.t. q_V = iso_{UV}(q_U)\) and \(q'_V = iso_{UV}(q'_U)\), then \(qv R _{strict} qV, R _{strict} q'_V\) and for every \(a \in sig(U)(q_U) = sig(V)(q_V), \eta_{(U, q_U, a)}(q'_U) = \eta_{(V, q_V, a)}(q'_V)\).

For every \(a \in sig(U)(q_U) \cap sig(V)(q_V)\) we have a unique \(\eta s.t C \mapsto a \mapsto \eta with C = config(U)(q_U) = config(V)(q_V)\) and \(\phi = created(U)(q_U)(a) = created(V)(q_V)(a)\). Hence for every configuration \(C' \in supp(\eta)\), it exists a unique pair \((q'_U, q'_V) \in reachable(U) \times reachable(V) s.t. C' = config(U)(q'_U) = config(V)(q'_V)\). Hence \(iso_{UV}(q'_U) = q'_V\) and furthermore \(\eta_{(U, q_U, a)}(q'_U) = \eta_{(V, q_V, a)}(q'_V) = \eta(C)\).

Everything is ready to construct the PCA-execution-matching, which is (j) the PCA-execution-matching induced by \(iso_{UV}\) and \(D_U\) (the set of discrete transition of \(U\)) and (jj) the PCA-execution-matching induced by \(iso_{UV}\) and \(D_V\) (the set of discrete transition of \(V\))

\(\Box\)

10 PCA CORRESPONDING W.R.T. PSIOA \(\mathcal{A}, \mathcal{B}\)

In the previous section we have shown that \(X_{\mathcal{A}}||E \) and \(\hat{\mathcal{A}}^{s\mathcal{w}}||\hat{\mathcal{X}} \) are linked by a strong PCA executions-matching as long as \(\mathcal{A}\) is not re-created by \(X_{\mathcal{A}}\). This also means that the probability distribution of \(X_{\mathcal{A}}||E\) is preserved by \(\hat{\mathcal{A}}^{s\mathcal{w}}||\hat{\mathcal{X}}\), as long as \(\mathcal{A}\) is not re-created by \(X_{\mathcal{A}}\). We can have the same reasoning to obtain a strong PCA executions-matching from \(X_{\mathcal{B}}||E \) and \(\hat{\mathcal{B}}^{s\mathcal{w}}||\hat{\mathcal{X}}\).

In this section we take an interest in PCA \(X_{\mathcal{A}}\) and \(X_{\mathcal{B}}\) that differ only on the fact that \(\mathcal{B}\) supplants \(\mathcal{A}\) in \(X_{\mathcal{B}}\). Hence, we recall the definitions of section 6. Then, we show that under slight assumptions, \(X_{\mathcal{A}} \) and \(X_{\mathcal{B}}\) are semantically equivalent (see theorem 10.13).

Combined with the result of previous section we will realise that we can obtain a strong PCA executions-matching from \((\mathcal{X} \mathcal{A}||E \) to \(\hat{\mathcal{A}}^{s\mathcal{w}}||\hat{\mathcal{Y}}\)) and \((\mathcal{X} \mathcal{B}||E \) to \(\hat{\mathcal{B}}^{s\mathcal{w}}||\hat{\mathcal{Y}}\)) where \(\mathcal{Y}\) is semantically equivalent to both \(\mathcal{X} \mathcal{B}||\hat{\mathcal{B}}\) and \(\mathcal{X} \mathcal{A}||\hat{\mathcal{A}}\). Hence if \(E' = E\)\(|\mathcal{Y}\) cannot distinguish \(\hat{\mathcal{A}}^{s\mathcal{w}}\) from \(\hat{\mathcal{B}}^{s\mathcal{w}}\), we will be able to show that \(E\) cannot distinguish \(X_{\mathcal{A}}\) from \(X_{\mathcal{B}}\) which will be the subject of sections 11 to finally prove the monotonicity of print-implementation.

\(\bowtie_{AB}-correspondence between two configurations\). We formalise the idea that two configurations are the same excepting the fact that the automaton \(\mathcal{B}\) supplants \(\mathcal{A}\) but with the same external signature. The next definition comes from [1].

**Definition 10.1** (\(\bowtie_{AB}\)-corresponding configurations). (see figure 30) Let \(\Phi \subseteq Autid\), and \(\mathcal{A}, \mathcal{B}\) be PSIOA identifiers. Then we define \(\Phi[\mathcal{B}/\mathcal{A}] = (\Phi \setminus \mathcal{A}) \cup \{\mathcal{B}\}\) if \(\mathcal{A} \in \Phi\), and \(\Phi[\mathcal{B}/\mathcal{A}] = \Phi\) if \(\mathcal{A} \notin \Phi\). Let \(C, D\) be configurations. We define \(C \bowtie_{AB} D\) iff (1) \(auts(D) = auts(C)|\mathcal{B}/\mathcal{A}\), (2) for every \(\mathcal{A}' \notin auts(C) \setminus |\mathcal{A}|: map(D)(\mathcal{A}') = map(C)(\mathcal{A}')\), and (3) \(ext(\mathcal{A})(s) = ext(\mathcal{B})(t)\) where \(s = map(C)(\mathcal{A}), t = map(D)(\mathcal{B})\). That is, in \(\bowtie_{AB}\)-corresponding configurations, the
SIOA other than \( \mathcal{A} \) and \( \mathcal{B} \) must be the same, and must be in the same state. \( \mathcal{A} \) and \( \mathcal{B} \) must have the same external signature. In the sequel, when we write \( \Psi = \Phi[\mathcal{B}/\mathcal{A}] \), we always assume that \( \mathcal{B} \not\in \Phi \) and \( \mathcal{A} \not\in \Psi \).

![Diagram](image.png)

Fig. 30. \( \llangle \mathcal{A}, \mathcal{B} \rrangle \) corresponding-configuration

Next lemma states that \( \llangle \mathcal{A}, \mathcal{B} \rrangle \)-corresponding configurations have the same external signature, which is quite intuitive when we see the figure 30.

**Proposition 1.** Let \( \mathcal{C}, \mathcal{D} \) be configurations such that \( \mathcal{C} \llangle \mathcal{A}, \mathcal{B} \rrangle \mathcal{D} \). Then \( \text{ext}(\mathcal{C}) = \text{ext}(\mathcal{D}) \).

**Proof.** The proof is in [1], section 6, p. 38. We write the proof here to be complete:

If \( \mathcal{A} \not\in \mathcal{C} \) then \( \mathcal{C} = \mathcal{D} \) by definition, and we are done. Now suppose that \( \mathcal{A} \in \mathcal{C} \), so that \( \mathcal{C} = (\mathcal{A} \cup \{\mathcal{A}\}, \mathcal{S}) \) for some set \( \mathcal{A} \) of PSIOA identifiers s. t. \( \mathcal{A} \not\in \mathcal{A} \), and let \( s = \text{S}(\mathcal{A}) \). Then, by definition 5.3 of attributes of configuration, \( \text{out}(\mathcal{C}) = (\bigcup_{\mathcal{A}_i \in \mathcal{A}} \text{out}(\mathcal{A}_i)) (\text{S}(\mathcal{A}_i)) \cup \text{out}(\mathcal{A})(s) \). From \( \mathcal{C} \llangle \mathcal{A}, \mathcal{B} \rrangle \mathcal{D} \) and definition, we have \( \mathcal{D} = (\mathcal{A} \cup \{\mathcal{B}\}, \mathcal{S}', \mathcal{B} \not\in \mathcal{S}(\mathcal{A}) \) and \( t = \mathcal{S}'(\mathcal{B}) \) such that \( \text{ext}(\mathcal{A})(s) = \text{ext}(\mathcal{B})(t) \). Hence \( \text{out}(\mathcal{A})(s) = \text{out}(\mathcal{B})(t) \) and \( \text{in}(\mathcal{A})(s) = \text{in}(\mathcal{B})(t) \). By definition 5.3 of configuration attributes, \( \text{out}(\mathcal{D}) = (\bigcup_{\mathcal{A}_j \in \mathcal{A}} \text{out}(\mathcal{A}_j)) (\text{S}(\mathcal{A}_j)) \cup \text{out}(\mathcal{B})(t) \). Finally, \( \text{out}(\mathcal{C}) = \text{out}(\mathcal{D}) \). Since \( \mathcal{S}' \) agrees with \( \mathcal{S} \) on all \( \mathcal{A}_i \in \mathcal{A} \), and \( t = \mathcal{S}'(\mathcal{B}) \) such that \( \text{ext}(\mathcal{A})(s) = \text{ext}(\mathcal{B})(t) \). We establish \( \text{ext}(\mathcal{C}) = \text{ext}(\mathcal{D}) \) in the same manner, and omit the repetitive details. Hence \( \text{ext}(\mathcal{C}) = \text{ext}(\mathcal{D}) \). \( \square \)

**Remark 3.** It is possible to have two configurations \( \mathcal{C}, \mathcal{D} \) s. t. \( \mathcal{C} \llangle \mathcal{A}, \mathcal{B} \rrangle \mathcal{D} \). That would mean that \( \mathcal{C} \) and \( \mathcal{D} \) only differ on the state of \( \mathcal{A} \) (or \( \mathcal{B} \)) that has even the same external signature in both cases \( \text{ext}(\mathcal{A})(s) = \text{ext}(\mathcal{A})(t) \), while we would potentially have \( \text{int}(\mathcal{A})(s) \neq \text{int}(\mathcal{A})(t) \).

The next lemma states that \( \llangle \mathcal{A}, \mathcal{B} \rrangle \)-corresponding configurations are equals if we omit the automata \( \mathcal{A} \) and \( \mathcal{B} \).

**Lemma 10.2 (Same configuration).** Let \( \mathcal{A}, \mathcal{B} \in \text{Autids} \). Let \( X_\mathcal{A}, X_\mathcal{B} \) be \( \mathcal{A} \)-fair and \( \mathcal{B} \)-fair PCA respectively, where \( X_\mathcal{A} \) never contains \( \mathcal{B} \) and \( X_\mathcal{B} \) never contains \( \mathcal{A} \). Let \( Y_\mathcal{A} = X_\mathcal{A} \setminus \{\mathcal{A}\}, Y_\mathcal{B} = X_\mathcal{B} \setminus \{\mathcal{B}\} \). Let \( (x_a, x_b) \in \text{states}(X_\mathcal{A}) \times \text{states}(X_\mathcal{B}) \) s. t. \( \text{conf}(X_\mathcal{A})(x_a) \llangle \mathcal{A}, \mathcal{B} \rrangle \text{conf}(X_\mathcal{B})(x_b) \).

Then \( \text{conf}(Y_\mathcal{A})(y_a) = \text{conf}(Y_\mathcal{B})(y_b) \).

**Proof.** By projection, we have \( \text{conf}(Y_\mathcal{A})(y_a) \llangle \mathcal{A}, \mathcal{B} \rrangle \text{conf}(Y_\mathcal{B})(y_b) \) with each configuration that does not contain \( \mathcal{A} \) nor \( \mathcal{B} \), thus for \( \text{conf}(Y_\mathcal{A})(y_a) \) and \( \text{conf}(Y_\mathcal{B})(y_b) \) contain the same set of automata ids (rule 1) of \( \llangle \mathcal{A}, \mathcal{B} \rrangle \) and map each automaton of this set to the same state (rule 2) of \( \llangle \mathcal{A}, \mathcal{B} \rrangle \). \( \square \)
same compartment of two PCA modulo $\mathcal{A}, \mathcal{B}$. In this paragraph we formalise the fact that two PCA have the same compartment, excepting for $\mathcal{B}$ that supplants $\mathcal{A}$.

First, we formalise the fact that two PCA create some PSIOA in the same manner, excepting for $\mathcal{B}$ that supplants $\mathcal{A}$. Here again, this definition comes from [1].

Definition 10.3 (Creation corresponding configuration automata). Let $X, Y$ be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that $X, Y$ are creation-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff
(1) $X$ never creates $\mathcal{B}$ and $Y$ never creates $\mathcal{A}$.
(2) Let $\beta \in \text{traces}^*(X) \cap \text{traces}^*(Y)$ a finite trace of both $X$ and $Y$, and let $\alpha \in \text{Execs}^*(X), \pi \in \text{Execs}^*(Y)$ a finite execution of both $X$ and $Y$ be such that $\text{trace}_\mathcal{A}(\alpha) = \text{trace}_\mathcal{A}(\pi) = \beta$. Let $x = \text{last}(\alpha), y = \text{last}(\pi)$, i.e., $x, y$ are the last states along $\alpha, \pi$, respectively. Then $\forall a \in \widetilde{\text{sig}}(X)(x) \cap \widetilde{\text{sig}}(Y)(y) : \text{created}(Y)(y)(a) = \text{created}(X)(x)(a)[\mathcal{B}/\mathcal{A}]$. Naturally $[\mathcal{B}/\mathcal{A}]$-corresponding sets of created automata are deprived of $\mathcal{A}$ and $\mathcal{B}$ respectively, they becomes equal, which is formalised in next lemma.

Lemma 10.4 (Same creation after projection). Let $\mathcal{A}, \mathcal{B} \in \text{Autids}$. Let $X_\mathcal{A}, X_\mathcal{B}$ be $\mathcal{A}$-fair and $\mathcal{B}$-fair PCA respectively, where $X_\mathcal{A}$ never contains $\mathcal{B}$ and $X_\mathcal{B}$ never contains $\mathcal{A}$ ($\mathcal{B} \notin \text{UA}(X_\mathcal{A})$ and $\mathcal{A} \notin \text{UA}(X_\mathcal{B})$). Let $Y_\mathcal{A} = X_\mathcal{A} \setminus \mathcal{A}$, $Y_\mathcal{B} = X_\mathcal{B} \setminus \mathcal{B}$. Let $(x_a, x_b) \in \text{states}(X_\mathcal{A}) \times \text{states}(X_\mathcal{B})$ and $\pi \in \text{Execs}^*(X_\mathcal{A})(x_a) \cap \text{Execs}^*(X_\mathcal{B})(x_b)$ such that $\text{created}(X_\mathcal{A})(x_a)(\pi) = \text{created}(X_\mathcal{B})(x_b)(\pi)$

Then $\text{created}(Y_\mathcal{A})(x_a)(\pi) = \text{created}(Y_\mathcal{B})(x_a)(\pi)$.}

Proof. By definition of PCA projection, we have $\text{created}(Y_\mathcal{B})(x_b)(\pi) = (\text{created}(X_\mathcal{B})(x_b)(\pi)) \setminus \mathcal{B} = (\text{created}(X_\mathcal{A})(x_a)(\pi)) \setminus \mathcal{A} = \text{created}(Y_\mathcal{A})(x_a)(\pi)$. $
$
Second, we formalise the fact that two PCA hide their actions in the same manner. The definition is strongly inspired by [1].

Definition 10.5 (Hiding corresponding configuration automata). Let $X, Y$ be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that $X, Y$ are hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff
(1) $X$ never creates $\mathcal{B}$ and $Y$ never creates $\mathcal{A}$.
(2) Let $\beta \in \text{traces}^*(X) \cap \text{traces}^*(Y)$, and let $\alpha \in \text{Execs}^*(X), \pi \in \text{Execs}^*(Y)$ be such that $\text{trace}_\mathcal{A}(\alpha) = \text{trace}_\mathcal{A}(\pi) = \beta$. Let $x = \text{last}(\alpha), y = \text{last}(\pi)$, i.e., $x, y$ are the last states along $\alpha, \pi$, respectively. Then $\text{hidden-actions}(Y)(y) = \text{hidden-actions}(X)(x)$.

Naturally if hidden actions of $\mathcal{A}\mathcal{B}$-corresponding states are equal, it remains true after respective deprivation of $\mathcal{A}$ and $\mathcal{B}$ which is formalised in next lemma.

Lemma 10.6 (Same hidden-actions after projection). Let $\mathcal{A}, \mathcal{B} \in \text{Autids}$. Let $X_\mathcal{A}, X_\mathcal{B}$ be $\mathcal{A}$-fair and $\mathcal{B}$-fair PCA respectively, where $X_\mathcal{A}$ never contains $\mathcal{B}$ and $X_\mathcal{B}$ never contains $\mathcal{A}$ ($\mathcal{B} \notin \text{UA}(X_\mathcal{A})$ and $\mathcal{A} \notin \text{UA}(X_\mathcal{B})$). Let $Y_\mathcal{A} = X_\mathcal{A} \setminus \mathcal{A}$, $Y_\mathcal{B} = X_\mathcal{B} \setminus \mathcal{B}$. Let $(x_a, x_b) \in \text{states}(X_\mathcal{A}) \times \text{states}(X_\mathcal{B})$, $y_a = X_\mathcal{A}\mu_\mathcal{A}(x_a), y_b = X_\mathcal{B}\mu_\mathcal{B}(x_b)$ s. t.
- $x_a\mathcal{A}\cap y_b$, i.e. $y_a\mathcal{A} \cap y_b$
- $\text{hidden-actions}(Y_\mathcal{A})(x_a) = \text{hidden-actions}(X_\mathcal{A})(x_a)$

Then $\text{hidden-actions}(Y_\mathcal{B})(x_b) = \text{hidden-actions}(Y_\mathcal{A})(y_a)$.
Proof. We note $C_{X_A} = \text{config}(X_A)(x_A)$, $C_{X_B} = \text{config}(X_B)(x_B)$, $C_{Y_A} = \text{config}(Y_A)(y_A)$, $C_{Y_B} = \text{config}(Y_B)(y_B)$. By assumption, $C_{X_A} \setminus \{A\} = C_{Y_A} = C_{Y_B} = C_{X_B} \setminus \{B\}$.

We note $h_{X_A} = \text{hidden-actions}(X_A)(x_A)$, $h_{X_B} = \text{hidden-actions}(X_B)(x_B)$, $h_{Y_A} = \text{hidden-actions}(Y_A)(y_A)$, $h_{Y_B} = \text{hidden-actions}(Y_B)(y_B)$. By assumption, $h_{X_A} = h_{X_B}$, while by construction, $h_{y_A} = h_{X_A} \setminus \text{pot-out}(X_A)(A)$ and $h_{Y_B} = h_{X_B} \setminus \text{pot-out}(X_B)(B)$.

Case 1: $\text{pot-out}(X_A)(A)(x_A) = \text{pot-out}(X_B)(B)(x_B)$, the result is immediate. Case 2: $\text{pot-out}(X_A)(A)(x_A) \cap h_{X_A} = \text{pot-out}(X_B)(B)(x_B) \cap h_{X_B} = \emptyset$, the result is immediate.

Case 3: Without loss of generality, we assume $\text{act} = \text{pot-out}(X_A)(A)(x_A) \cap h_{X_A} \neq \emptyset$. For every $C \in \text{auts}(C_{Y_B})$, $C \in \text{auts}(C_{Y_A})$ since $C_{Y_A} = C_{Y_B}$ and $C \in \text{auts}(C_{X_A})$ since $C_{Y_A} = C_{X_A} \setminus \{A\}$. By compatibility of $C_{X_A}$, $\text{pot-out}(X_A)(A)(x_A) \cap \text{pot-out}(X_A)(C)(x_a) = \emptyset$.

Case 3a) $B \notin \text{auts}(C_{X_B})$, which means both i) $\text{act} \subset h_{X_B}$, ii) $\text{act} \cap \text{out}(C_{X_B}) = \emptyset$ and iii) $h_{X_B} \subset \text{out}(C_{X_B})$ which is impossible. Thus we only consider

Case 3b) $B \in \text{auts}(C_{X_B})$. Since j) for every $C \in \text{auts}(C_{Y_B})$, $\text{pot-out}(X_A)(A)(x_A) \cap \text{pot-out}(X_A)(C)(x_a) = \emptyset$ and jj) $h_{X_B} \subset \text{out}(C_{X_B})$, we have $\text{act} \subset \text{pot-out}(X_B)(B)(x_B)$.

For symmetrical reason, we have both $\text{pot-out}(X_A)(A)(x_a) \cap h_{X_A} \subset \text{pot-out}(X_B)(B)(x_B)$ and $\text{pot-out}(X_B)(B)(x_B) \cap h_{X_B} \subset \text{pot-out}(X_A)(A)(x_a)$, which means $h_{X_A} \setminus \text{pot-out}(X_B)(B)(x_B) = h_{X_B} \setminus \text{pot-out}(X_B)(B)(x_B)$ and ends the proof.

Now we are ready to define corresponding PCA w.r.t. PSIOA $A$, $B$, that is two PCA $X_A$ and $X_B$ that differ only on the fact that $B$ supplants $A$ in $X_B$. Some additional assumptions are added to ensure monotonicity later. This definition is still inspired by definitions of [1].

Definition 10.7 (corresponding w.r.t. $A$, $B$). Let $A, B \in \text{Autids}$, $X_A$ and $X_B$ be PCA we say that $X_A$ and $X_B$ are corresponding w.r.t. $A$, $B$, if they verify:

- $\text{config}(X_A)(q_{X_A}) \prec_A \text{config}(X_B)(q_{X_B})$.
- $X_A$ never contains $B$ ($B \notin \text{UA}(X_A)$), while $X_B$ never contains $A$ ($A \notin \text{UA}(X_B)$).
- $X_A, X_B$ are creation-corresponding w.r.t. $A, B$.
- $X_A, X_B$ are hiding-corresponding w.r.t. $A, B$.
- $X_A$ (resp. $X_B$) is a $A$-conservative (resp. $B$-conservative) PCA.
- (No exclusive creation from $A$ and $B$)
  - $\forall q_{X_A} \in \text{states}(X_A)$, for every action $\text{act}$ $A$-exclusive, $\text{created}(X_A)(q_{X_A})(\text{act}) = \emptyset$ and similarly
  - $\forall q_{X_B} \in \text{states}(X_B)$, for every action $\text{act}'$ $B$-exclusive, $\text{created}(X_B)(q_{X_B})(\text{act}') = \emptyset$

Equivalent transitions to obtain semantic equivalence after projection. In this last paragraph of the section, we show that if two PCA $X_A, X_B$ are corresponding w.r.t. $A$ and $B$, then there respective projection $Y_A = X_A \setminus \{A\}$ and $Y_B = X_B \setminus \{B\}$ are semantically equivalent. To do so, we use notions of equivalent transitions. The idea is to recursively show that any corresponding executions of $Y_A$ and $Y_B$ lead to strictly equivalent transitions to finally build the complete bijective PCA executions-matching from $Y_A$ to $Y_B$.

We start by defining equivalent transitions.

Definition 10.8 (configuration-equivalence and strict-equivalence between two distributions). Let $K, K'$ be PCA and $(\eta, \eta') \in \text{Disc}(\text{states}(K)) \times \text{Disc}(\text{states}(K'))$. 

\[\text{...}\]
• We say that $\eta$ and $\eta'$ are config-equivalent, noted $\eta \xleftarrow{\text{conf}} \eta'$, if it exists $f : \text{states}(K) \rightarrow \text{states}(K')$ s. t.

$\eta \xleftarrow{f} \eta'$ with $\forall q'' \in \text{supp}(\eta), q'' \in \text{conf}(f(q''))$.

• If additionally, $\forall q'' \in \text{supp}(\eta), q'' \in \text{strict}(f(q''))$, then we say that $\eta$ and $\eta'$ are strictly-equivalent, noted $\eta \xleftarrow{\text{strict}} \eta'$.

Basically, equivalent transitions are transitions where the states with non-zero probability to be reached are mapped by a bijective function that preserves i) measure of probability and ii) configuration. A stricter version preserves also iii) future created automata and hidden-actions.

The next lemma states that if we take two corresponding transitions from strict equivalent states, then we obtain configuration equivalent transitions.

**Lemma 10.9.** (strictly-equivalent states implies config-equivalent transition) Let $K, K'$ be PCA and $(q, q') \in \text{states}(K) \times \text{states}(K')$ strictly-equivalent, i.e. $q R_{\text{strict}} q'$. Let $a \in \text{sig}(K)(q) = \text{sig}(K')(q')$ and $(q, a, \eta(q,a)), (q', a, \eta(q',a)) \in \text{dtrans}(K) \times \text{dtrans}(K')$. Then $\eta(q,a)$ and $\eta(q',a)$ are config-equivalent, i.e. $\exists f : \text{states}(K) \rightarrow \text{states}(K')$ s. t.

$\eta \xleftarrow{\text{conf}} f \eta'$.

**Proof.** This is the direct consequence of constraint 2 and 3 of definition 5.14 of PCA. We note $C = \text{config}(K)(q) = \text{config}(K')(q')$ and $\phi = \text{created}(K)(q)(a) = \text{created}(K')(q')(a)$. By constraint 2, applied to $K$, it exists $\eta$ s. t.

$\eta(q,a) \xleftarrow{f} \eta$ with $f^K = \text{config}(K)$ and $\text{config}(K)(q) \xrightarrow{a} \text{created}(K)(q)(a) \eta$. By constraint 2, applied to $K'$, it exists $\eta'$ s. t. $\eta(q',a) \xleftarrow{f} \eta'$ with $f^{K'} = \text{config}(K')$ and $\text{config}(K')(q') \xrightarrow{a} \text{created}(K')(q')(a) \eta'$.

Since $q R_{\text{strict}} q'$, $C \equiv \text{config}(K)(q) = \text{config}(K')(q')$ and $\phi \equiv \text{created}(K)(q)(a) = \text{created}(K')(q')(a)$.

Hence $C \xrightarrow{a} \phi$ and $C \xrightarrow{a} \phi'$ which means $\eta = \eta'$.

So $\eta(q,a) \xleftarrow{f} \eta(q',a)$ with $f = (f^{K'})^{-1} \circ f^K$ where $f$ (resp. $f^{K'}$, resp. $f^K$) is the restriction of $f$ (resp. $f^{K'}$, resp. $f^K$) on $\text{supp}(\eta(q,a))$ (resp. $\text{supp}(\eta(q',a))$, resp. $\text{supp}(\eta(q,a))$).

Thus, for every $(\tilde{q}, \tilde{q}') \in \text{supp}(\eta(q,a)) \times \text{supp}(\eta(q',a))$ s. t. $\tilde{q}' = f(\tilde{q})$. $f^K(\tilde{q}) = f^{K'}(\tilde{q}')$, that is $\text{config}(K)(\tilde{q}) = \text{config}(K')(\tilde{q}')$, i.e. $\tilde{q} R_{\text{conf}} \tilde{q}'$.

Hence $\eta(q,a) \xleftarrow{\text{conf}} f \eta(q',a)$ which ends the proof.

Now we start a sequence of lemma (from lemma 10.10 to lemma 10.12) to finally show in theorem 10.13 that if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w. r. t. $\mathcal{A}, \mathcal{B}$ then $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically-equivalent.

The next lemma shows that we can always construct an execution $\hat{\alpha}_X \in \text{Execs}(X)$ from an execution $\alpha_Y \in \text{Execs}(Y)$ with $Y = X \setminus \{\mathcal{A}\}$ that preserves the trace.

**Lemma 10.10 (Execs(X \ {A}) can be obtained by Execs(X)).** Let $\mathcal{A} \in \text{Autids}, X a \mathcal{A}$-fair PCA, $Y = X \setminus \{\mathcal{A}\}$.

Let $\alpha_Y = q^0_Y, a^1, q^1_Y, ..., q^n_Y \in \text{Execs}(Y)$. Then it exists, $\hat{\alpha}_X = q^0_X, a^1, q^1_X, ..., q^n_X \in \text{Execs}(X)$ s. t. $\forall i \in [0, n], q^i_Y = \mu^A(q^i_X)$.

**Proof.** By induction on the size $s = |\alpha_Y|$ of prefix $\alpha^s_Y = q^0_Y, a^1, q^1_Y, ..., q^s_Y$.

**Basis ($|\alpha^0_Y| = 0$):** By definition 8.11, $q^0_Y = X, \mu^A(q^0_X)$

**Induction:** let assume the proposition is true for prefix $\alpha^s_Y = q^0_Y, a^1, q^1_Y, ..., q^s_Y$ with $s < |\alpha_Y|$. We will show it is true for $\alpha^{s+1}_Y$. We have $q^s_Y = X, \mu^A(q^s_X)$. By construction of $\text{dtrans}(Y)$ provided by definition 8.11, it exists $\eta(q^s_X, a^{s+1}_Y) \in$
Lemma 10.11 (After projection, configuration-equivalence obtain after same trace implies strict equivalence). Let \( X_A \) and \( X_B \) be two PCA corresponding w. r. t. \( A, B \). Let \( Y_A = X_A \setminus \{ A \} \) and \( Y_B = X_B \setminus \{ B \} \). Let \((a_{Y_A}, \pi_{Y_B}) \in \text{Execs}(Y_A) \times \text{Execs}(Y_B)\) with \( \text{Istate}(a_{Y_A}) = q_{Y_A} \) and \( \text{Istate}(\pi_{Y_B}) = q_{Y_B} \). If

- \( q_{Y_A} R_{\text{conf}} q_{Y_B} \)
- \( \text{trace}(a_{Y_A}) = \text{trace}(\pi_{Y_B}) = \beta \),

then \( q_{Y_A} R_{\text{strict}} q_{Y_B} \).

Proof. By lemma 10.10, it exists \((\tilde{a}_{X_A}, \tilde{\pi}_{X_B}) \in \text{Execs}(X_A) \times \text{Execs}(X_B)\) s. t. (i) \( \text{trace}(\tilde{a}_{X_A}) = \text{trace}(a_{Y_A}) = \text{trace}(\pi_{Y_B}) \) and (ii) \( q_{Y_A} = X_A \mu^A_a(\tilde{q}_{X_A}) \) and \( q_{Y_B} = X_B \mu^B_a(\tilde{q}_{X_B}) \) where \( \tilde{q}_{X_A} = \text{Istate}(\tilde{\pi}_{X_B}) \) and \( \tilde{q}_{X_B} = \text{Istate}(\tilde{\pi}_{X_B}) \).

Since \( \text{trace}(\tilde{a}_{X_A}) = \text{trace}(\pi_{Y_B}) \), we have j) \( \text{hidden-actions}(X_A)(\tilde{q}_{X_A}) = \text{hidden-actions}(X_B)(\tilde{q}_{X_B}) \) by hiding-correspondence of definition 6.3 and jj) \( \forall a \in \text{sig}(X_A)(\tilde{q}_{X_A}) \cup \text{supp}(X_B)(\tilde{q}_{X_B}), \text{created}(X_A)(\tilde{q}_{X_A})(a) = \text{created}(X_B)(\tilde{q}_{X_B})(a) \).

By lemma 10.6 we have (*) \( \text{hidden-actions}(Y_A)(\tilde{q}_{Y_A}) = \text{hidden-actions}(Y_B)(\tilde{q}_{Y_B}) \), and by lemma 10.4 we have (**) \( \forall a \in \text{sig}(Y_A)(\tilde{q}_{Y_A}) = \text{sig}(Y_B)(\tilde{q}_{Y_B}) \).

If we combine the definition \( q_{Y_A} R_{\text{conf}} q_{Y_B} \) with (*) and (**), we obtain \( q_{Y_A} R_{\text{strict}} q_{Y_B} \), which ends the proof. \( \square \)

Finally, the next lemma states that, after projection, two configuration-equivalent states obtain via executions with the same trace lead necessarily to strictly equivalent transitions.

Lemma 10.12 (After projection, configuration-equivalence obtain after same trace implies strict equivalent transitions). Let \( X_A \) and \( X_B \) be two PCA corresponding w. r. t. \( A, B \). Let \( Y_A = X_A \setminus \{ A \} \) and \( Y_B = X_B \setminus \{ B \} \). Let \((a_{Y_A}, \pi_{Y_B}) \in \text{Execs}(Y_A) \times \text{Execs}(Y_B)\) with \( \text{Istate}(a_{Y_A}) = q_{Y_A} \) and \( \text{Istate}(\pi_{Y_B}) = q_{Y_B} \). If

- \( q_{Y_A} R_{\text{conf}} q_{Y_B} \)
- \( \text{trace}(a_{Y_A}) = \text{trace}(\pi_{Y_B}) = \beta \),

then for every \( a \in \text{sig}(Y_A)(\tilde{q}_{Y_A}) = \text{sig}(Y_B)(\tilde{q}_{Y_B}) \), \( \eta(Y_A, q_{Y_A}, a) \) and \( \eta(Y_B, q_{Y_B}, a) \) are strictly equivalent, i. e. \( \exists f : \text{states}(K) \rightarrow \text{states}(K') \) s. t. \( \eta \xrightarrow{f\text{ strict}} \eta' \).

Proof. By previous lemma 10.11, \( q_{Y_A} \) and \( q_{Y_B} \) are strictly equivalent. Thus by previous lemma 10.9, it exists \( f \) s. t. \( \eta(Y_A, q_{Y_A}, a) \xrightarrow{f\text{ conf}} \eta(Y_B, q_{Y_B}, a) \). Let two corresponding states \((q'_{Y_A}, q'_{Y_B}) \in \text{supp}(\eta(Y_A, q_{Y_A}, a)) \times \eta(Y_B, q_{Y_B}, a)\) s. t. \( f(q'_{Y_A}) = q'_{Y_B} \). We have \( q'_{Y_A} R_{\text{conf}} q'_{Y_B} \). Furthermore, since \( q_{Y_A} R_{\text{strict}} q_{Y_B} \), \( \text{sig}(Y_A)(q_{Y_A}) = \text{sig}(Y_B)(q_{Y_B}) \), namely \( \text{ext}(Y_A)(q_{Y_A}) = \text{ext}(Y_B)(q_{Y_B}) \), which means \( \text{trace}(\eta(Y_A, q_{Y_A}, a)) = \text{trace}(\eta(Y_B, q_{Y_B}, a)) \). So we can reapply previous lemma to obtain \( q'_{Y_A} R_{\text{strict}} q'_{Y_B} \), which ends the proof. \( \square \)
Now we can finally show that if $X_\mathcal{A}$ and $X_\mathcal{B}$ are corresponding w. r. t. $\mathcal{A}$, $\mathcal{B}$ then $X_\mathcal{A} \setminus \{\mathcal{A}\}$ and $X_\mathcal{B} \setminus \{\mathcal{B}\}$ are semantically-equivalent which was the main aim of this subsection.

**Theorem 10.13 (X_\mathcal{A} AND X_\mathcal{B} CORRESPONDING W. R. T. \mathcal{A}, \mathcal{B} IMPLIES X_\mathcal{A} \setminus \{\mathcal{A}\} AND X_\mathcal{B} \setminus \{\mathcal{B}\} SEMANTICALLY-EQUIVALENT).** Let $X_\mathcal{A}$ and $X_\mathcal{B}$ be two PCA corresponding w. r. t. $\mathcal{A}$, $\mathcal{B}$. Let $Y_\mathcal{A} = X_\mathcal{A} \setminus \{\mathcal{A}\}$ and $Y_\mathcal{B} = X_\mathcal{B} \setminus \{\mathcal{B}\}$.

The PCA $Y_\mathcal{A}$ and $Y_\mathcal{B}$ are semantically-equivalent.

**Proof.** We recursively construct a strong complete bijective PCA executions-matching $(f_s, f_{s,trans}, f_{s,exec})$ where $f_s : reachable_{\leq s}(Y_\mathcal{A}) \rightarrow reachable_{\leq s}(Y_\mathcal{B})$ and $f_{s,exec} : \{a \in Execs(Y_\mathcal{A})||a|| \leq s\} \rightarrow \{\pi \in Execs(Y_\mathcal{B})||\pi|| \leq s\}$ s. t. $f_{s,exec}(a) = \pi$ implies $\text{Istate}(a) \text{Restrict}(\text{Istate}(\pi))$.

**Base**: $s = 0$, $reachable_{\leq 0}(Y_\mathcal{A}) = \{\bar{q}_X\}$, while $reachable_{\leq 0}(Y_\mathcal{B}) = \{\bar{q}_X\}$.

By definition 6.16 of corresponding automata $config(X_\mathcal{A})(\bar{q}_X) \equiv \mathcal{A} \mathcal{B} config(X_\mathcal{B})(\bar{q}_X)$, while $(\bar{q}_{Y_\mathcal{A}}, \bar{q}_{Y_\mathcal{B}}) = (X_\mathcal{A}, \mu^\mathcal{A}(\bar{q}_X), X_\mathcal{B}, \mu^\mathcal{B}(\bar{q}_X))$ by definition 8.11 of PCA projection, which gives $\bar{q}_{Y_\mathcal{A}} \text{Restrict} \bar{q}_{Y_\mathcal{B}}$ by lemma 10.2. Moreover $trace_{\mathcal{A}}(\bar{q}_{Y_\mathcal{A}}) = trace_{\mathcal{B}}(\bar{q}_{Y_\mathcal{B}}) = \lambda (\lambda$ denotes the empty sequence$)$. Thus we can apply lemma 10.11 to obtain $\bar{q}_{Y_\mathcal{A}} \text{Restrict} \bar{q}_{Y_\mathcal{B}}$. We construct $f_0(\bar{q}_{Y_\mathcal{A}}) = \bar{q}_{Y_\mathcal{B}}$, $f_{s,exec}(\bar{q}_{Y_\mathcal{A}}) = \bar{q}_{Y_\mathcal{B}}$. Clearly $f_0$ is a bijection from $reachable_0(Y_\mathcal{A})$ to $reachable_0(Y_\mathcal{B})$, while $f_{s,exec}$ is a bijection from $Execs_0(Y_\mathcal{A})$ to $Execs_0(Y_\mathcal{B})$.

**Induction:** We assume the result to be true for an integer $s \in \mathbb{N}$ and we will show it is then true for $s + 1$. Let $Execs_s(Y_\mathcal{A}) = \{a \in Execs(Y_\mathcal{A})||a|| = s\}$ and $Execs_s(Y_\mathcal{B}) = \{a \in Execs(Y_\mathcal{B})||\pi|| = s\}$.

We can build $f_{s+1}(resp. f_{s+1})$ s. t. $\forall q \in reachable_{\leq s}, f_{s+1}(q) = f_s(q)$ (resp. s. t. $\forall a \in Execs_{\leq s}(Y_\mathcal{A}) f_{s,exec}(a) = f_{s,exec}(a)$) and $\bar{q}_{Y_\mathcal{A}} \in reachable_{\leq s+1}, f_{s+1}(q')$ (resp. $\forall a^{\alpha,j} \in Execs_{\leq s+1}(Y_\mathcal{A}), f_{s,exec}(a^{\alpha,j})$) is built as follows:

We note $a^{\alpha,j} = a^{-\alpha,j} \bar{q}_{Y_\mathcal{A}} a q'_{Y_\mathcal{A}} (q'_{Y_\mathcal{A}} = \text{Istate}(a^{\alpha,j}))$. We note $\pi_{Y_\mathcal{B}} = f_{s,exec}(a^{\alpha,j})$. By induction assumption, $\pi_{Y_\mathcal{B}} \text{Restrict} \pi_{Y_\mathcal{B}}$ with $\bar{q}_{Y_\mathcal{A}} = \text{Istate}(a^{\alpha,j})$ and $\pi_{Y_\mathcal{B}} = \text{Istate}(\pi_{Y_\mathcal{B}})$. Hence $\text{sig}(Y_\mathcal{A})(q'_{Y_\mathcal{A}}) = \text{sig}(Y_\mathcal{B})(q'_{Y_\mathcal{B}})$ and by previous lemma 10.12, for every $a \in \text{sig}(Y_\mathcal{A})(q'_{Y_\mathcal{A}}) = \text{sig}(Y_\mathcal{B})(q'_{Y_\mathcal{B}}), \exists q'_{Y_\mathcal{A}} \in \text{Istate}(a^{\alpha,j})$.

Hence, we define $f_{s,exec} : a^{\alpha,j} \rightarrow a^{-\alpha,j} \bar{q}_{Y_\mathcal{A}} a q'_{Y_\mathcal{A}} (q'_{Y_\mathcal{A}} = \text{Istate}(a^{\alpha,j}))$. While $f_{s+1}$ is naturally defined via $f_{s,exec}$, i. e. for every $q'_{Y_\mathcal{A}} \in reachable_{\leq s+1}(Y_\mathcal{A})$, we note $a^{\alpha,j} \in Execs_{\leq s+1}(Y_\mathcal{A})$ s. t. $\text{Istate}(a^{\alpha,j}) = q'_{Y_\mathcal{A}}$ and $f_{s+1}(q'_{Y_\mathcal{A}}) = \bar{q}_{Y_\mathcal{A}}$. We finally define $f_{s,exec} : q'_{Y_\mathcal{A}} a^{\alpha,j} \rightarrow f_0(q'_{Y_\mathcal{A}} a^{\alpha,j} f_n(q'_{Y_\mathcal{A}}))$, $f : q \rightarrow f_n(q)$ where $q = \text{Istate}(q'_{Y_\mathcal{A}} a^{\alpha,j} f_n(q'_{Y_\mathcal{A}}))$ and $f_{s,exec} : \{a, \pi | \text{Istate}(a) \rightarrow \text{Istate}(\pi)\}$.

Clearly $(f, f_{s,exec}, f_{s,exec})$ is strong for every pair $(q_{Y_\mathcal{A}}, q_{Y_\mathcal{B}})$, s. t. $f(q_{Y_\mathcal{A}}) = q_{Y_\mathcal{B}}$, $q_{Y_\mathcal{A}}, q_{Y_\mathcal{B}}$. Moreover, $(f, f_{s,exec}, f_{s,exec})$ is complete since $\text{dom}(f) = reachable(Y_\mathcal{A}) = \text{states}(Y_\mathcal{A})$.

Finally, the bijectivity of $f_{s,exec}$ is given by the inductive bijective construction.

Hence $(f, f_{s,exec}, f_{s,exec})$ is strong complete bijective PCA executions-matching from $Y_\mathcal{A}$ to $Y_\mathcal{B}$ which ends the proof.

\[\square\]

**11 TOP/DOWN CORRESPONDING CLASSES (BIS)**

In previous section 10, we have shown in theorem 10.13 that if $X_\mathcal{A}$ and $X_\mathcal{B}$ are corresponding w. r. t. $\mathcal{A}$ and $\mathcal{B}$ (in the sense of definition 6.16), then $Y_\mathcal{A} = X_\mathcal{A} \setminus \{\mathcal{A}\}$ and $Y_\mathcal{B} = X_\mathcal{B} \setminus \{\mathcal{B}\}$ are semantically equivalent. We can note $Y$ an arbitrary PCA semantically equivalent with both $Y_\mathcal{A}$ and $Y_\mathcal{B}$.

In section 9, we have shown in theorem 9.19 that for every PCA $E$ environment of both $X_\mathcal{A}$ and $X_\mathcal{B}$, $X_\mathcal{A}|E$ and $\overline{X_\mathcal{A}}|E$ (resp. $X_\mathcal{B}|E$ and $\overline{X_\mathcal{B}}|E$) are linked by a PCA executions-matching
It is time to combine these two results to realize that for every PCA \( E \) environment of both \( X_A \) and \( X_B \), \( X_A || E \) and \( \tilde{A}^* || E' \) (resp. \( X_B || E \) and \( \tilde{B}^* || E' \)) are linked by a PCA executions-matching where \( E' = E | Y \).

Hence (*) if \( E' \) cannot distinguish \( \tilde{A}^* \) from \( \tilde{B}^* \), we will be able to show that \( E \) cannot distinguish \( X_A \) from \( X_B \).

In this section, we formalise (*) in theorem 11.25 of monotonicity of implementation relation. However, some assumptions are required to reduce the implementation of \( X_B \) by \( X_A \) into implementation of \( B \) by \( A \). These are all

### 11.1 Creation-oblivious scheduler

Here we recall the definition of creation-oblivious scheduler (already introduced in subsection 6.4), that does not take into account previous internal actions of a particular sub-automaton to output its probability over transitions to trigger.

We start by defining \emph{strict oblivious-schedulers} that output the same transition with the same probability for pair of execution fragments that differ only by prefixes in the same class of equivalence. This definition is inspired by the one provided in the thesis of Segala, but is more restrictive since we require a strict equality instead of a correlation (section 5.6.2 in [8]).

**Definition 11.3 (creation-oblivious scheduler).** Let \( X \) be a PCA or a PSIOA, let \( \sigma \in \text{scheduler}(\tilde{W}) \) and let \( \equiv \) be an equivalence relation on \( \text{Frag}_s^* (\tilde{W}) \) verifying \( \forall \hat{a}_1, \hat{a}_2 \in \text{Frag}_s^* (\tilde{W}) \) s. t. \( \hat{a}_1 \equiv \hat{a}_2 \), \( \text{Istate}(\hat{a}_1) = \text{Istate}(\hat{a}_2) \). We say that \( \sigma \) is \( (\equiv) \)-strictly oblivious if \( \forall \hat{a}_1, \hat{a}_2, \hat{a}_3 \in \text{Frag}_s^* (\tilde{W}) \) s. t. 1) \( \hat{a}_1 \equiv \hat{a}_2 \) and 2) \( f\text{state}(\hat{a}_3) = \text{Istate}(\hat{a}_2) = \text{Istate}(\hat{a}_1) \), then \( \hat{\sigma}(\hat{a}_1, \hat{a}_3) = \hat{\sigma}(\hat{a}_2, \hat{a}_3) \).

Now we define the relation of equivalence that defines our subset of creation-oblivious schedulers. Intuitively, two execution fragments ending on \( A \) creation are in the same equivalence class if they differ only in terms of internal actions of \( A \).

**Definition 11.2 \((\hat{a} \equiv^c_A \hat{a}') (\text{recall})\).** Let \( \tilde{A} \) be a PSIOA, \( \tilde{W} \) be a PCA, \( \forall \hat{a}, \hat{a}' \in \text{Frag}_s^* (\tilde{W}) \), we say \( \hat{a} \equiv^c_A \hat{a}' \) iff:

1. \( \hat{a}, \hat{a}' \) both ends on \( A \)-creation.
2. \( \hat{a} \) and \( \hat{a}' \) differ only in the \( A \)-exclusive actions and the states of \( A \), i.e., \( \mu(\hat{a}) = \mu(\hat{a}') \) where \( \mu(\hat{a}) = \tilde{q}^{n_1}a_1^{q_1}...a^nq^n \) ∈ \( \text{Frag}_s^* (\tilde{W}) \) is defined as follows:
   - remove the \( A \)-exclusive actions
   - replace each state \( \tilde{q}^j \) by its configuration \( \text{Config}(\tilde{W})(\tilde{q}) = (A', S') \)
   - replace each configuration \( (A', S') \) by \( (A', S') \setminus \{A\} \)
   - replace the (non-alternating) sequences of identical configurations (due to \( A \)-exclusiveness of removed actions) by one unique configuration.
3. \( \text{trace}(\hat{a}) = \text{trace}(\hat{a}') \),
4. \( \text{Istate}(\hat{a}_1) = \text{Istate}(\hat{a}_2) \)

We can remark that the items 4 can be deduced from 1 and 2 if \( X \) is configuration-conflict-free. We can also remark that if \( \tilde{W} \) is a \( A \)-conservative PCA, we can replace \( \mu(\hat{a}) = \mu(\hat{a}') \), by \( \mu^A(\hat{a}) \uparrow (\tilde{W} \setminus \{A\}) = \mu^A(\hat{a}') \uparrow (\tilde{W} \setminus \{A\}) \) but we want to be as general as possible for next definition of \emph{creation oblivious scheduler}.

**Definition 11.3 (creation-oblivious scheduler).** Let \( \tilde{A} \) be a PSIOA, \( \tilde{W} \) be a PCA, \( \sigma \in \text{scheduler}(\tilde{W}) \). We say that \( \sigma \) is \( A \)-creation oblivious if it is \( (\equiv^c_A) \)-strictly oblivious.
We say that \( \hat{\sigma} \) is creation-oblivious if it is \( \mathcal{A} \)-creation oblivious for every sub-automaton \( \mathcal{A} \) of \( \hat{W} \) \((\mathcal{A} \in \bigcup q \text{states}(\hat{W}) \text{ auts}(\text{config}(\hat{W})(q)))\). We note \text{CrOB} the function that maps every PCA \( \hat{W} \) to the set of creation-oblivious schedulers of \( \hat{W} \). If \( \hat{W} \) is not a PCA but a PSIOA, \text{CrOB}(\hat{W}) = \text{ schedulers}(\hat{W}) \).

If \( \hat{W} \) is \( \mathcal{A} \)-conservative and \( \hat{\sigma} \) is \( \mathcal{A} \)-creation oblivious, we note \text{oblivious}_{\mathcal{A}, \beta, \epsilon}(\hat{\sigma}) \) (and usually \( \hat{\sigma}_{\mathcal{A}, \beta, \epsilon} \) when it is clear in the context) the (unique by definition) scheduler s. t. for every \( \hat{\alpha}, \hat{\alpha}' \in \text{Frags}^* (\hat{W}) \) with i) \( \hat{\alpha} \) ending on \( \mathcal{A} \)-creation , ii) \( \hat{\mu}_{\mathcal{A}} (\hat{\alpha}) \uparrow (\hat{W} \setminus \{\mathcal{A}\}) = e \), iii) \( \text{trace}(\hat{\alpha}) = \beta \) and iv) \( f\text{state}(\hat{\alpha}^\prime) = l\text{state}(\hat{\alpha}) \), then \( \hat{\sigma}_{\mathcal{A}, \beta, \epsilon}(\hat{\alpha}^\prime) = \hat{\sigma}(\hat{\alpha}^\prime \hat{\alpha}') \). Let us note that \( l\text{state}(\hat{\alpha}) \) is entirely defined with i) and ii). We remark that if such an execution fragment \( \hat{\sigma} \in \text{Frags}^* (\hat{W}) \) verifying i), ii) and iii) exists, then \( \hat{\sigma}_{\mathcal{A}, \beta, \epsilon} = \hat{\sigma}_{\bar{\alpha}, \beta} \) the sub-scheduler conditioned by \( \hat{\sigma} \) and \( \bar{\alpha} \) in the sense of definition 11.4 stated immediately below.

**Definition 11.4 (conditioned scheduler).** Let \( \mathcal{A} \) be a PSIOA, \( \sigma \in \text{ schedulers}(\mathcal{A}) \) and let \( a_1 \in \text{Frags}^* (\mathcal{A}) \). We note \( \sigma_{\left| a_1 \right|} : \{a_2 \in \text{Frags}^* (\mathcal{A}) \mid f\text{state}(a_2) = l\text{state}(a_1) \} \rightarrow \text{SubDisc} (d\text{trans}(\mathcal{A})) \) the sub-scheduler conditioned by \( \sigma \) and \( a_1 \) that verifies \( a_2 \in \text{Frags}^* (\mathcal{A}) \), \( f\text{state}(a_2) = l\text{state}(a_1) \) \( \pm q_{12} \). Then

\[
\epsilon_{\sigma, a_0} (\text{C}_{\text{a}_1^{-1}a_2}) := \begin{cases} 
\epsilon_{\sigma, a_0} (\text{C}_{a_1}) \cdot \epsilon_{\sigma_0, q_{12}} (\text{C}_{a_2}) & \text{if } a_1 \notin a_0 \\
\epsilon_{\sigma_{\left| a_1 \right|}, a_0} (\text{C}_{a_2}) & \text{if } a_0 = a_1^{-1} a_2' 
\end{cases}
\]

**Proof.** We note \( a_{12} = a_1^{-1} a_2 \).

1. \( a_1 \notin a_0 \):
   a) \( a_1 \notin a_0 \) and \( a_0 \notin a_1 \):
   This implies \( a_{12} \notin a_0 \) and \( a_0 \notin a_{12} \) thus \( \epsilon_{\sigma, a_0} (\text{C}_{a_1^{-1}a_2}) = \epsilon_{\sigma, a_0} (\text{C}_{a_1}) = 0 \) which ends the proof.

b) \( a_0 \leq a_1 \):
   This implies \( a_0 \leq a_{12} \) By induction on size \( s \) of \( a_{12} \). Basis: \( s = 0 \), i.e. \( a_2 = l\text{state}(a_1) = q_{12} \). Thus, we meet the second case of definition of \( \epsilon_{\sigma_{\left| a_1 \right|}, q_{12}} (\text{C}_{a_2}) \): \( a_2 \leq q_{12} \), which means \( \epsilon_{\sigma_{\left| a_1 \right|}, q_{12}} (\text{C}_{a_2}) = 1 \) and terminates the basis. Induction: We assume the result to be true up to size \( s \in \mathbb{N} \) and we want to show that it is still true for size \( s + 1 \). Let \( a_2 \in \text{Frags}^* (\mathcal{A}) \), \( f\text{state}(a_2) = l\text{state}(a_1) \pm q_{12} \) with \( |a_2| = s + 1 \). We note \( a_2 = a_2' \tilde{q} q_{12} \) and \( a_{12}' = a_1^{-1} a_2' \). We have \( |a_{12}'| = s \) and \( a_0 \leq a_{12}' \). By definition we have \( \epsilon_{\sigma_{\left| a_1 \right|}, q_{12}} (\text{C}_{a_2'}) = \epsilon_{\sigma_{\left| a_1 \right|}, q_{12}} (\text{C}_{a_2}) \cdot \sigma (a_2') (\eta (\mathcal{A} q_{12}) \cdot \eta (\mathcal{A} q_{12}) (q) \text{ and by induction assumption}, \epsilon_{\sigma, a_0} (\text{C}_{a_2'}) = \epsilon_{\sigma, a_0} (\text{C}_{a_2}) \cdot \sigma (a_{12}') (\eta (\mathcal{A} q_{12}) \cdot \eta (\mathcal{A} q_{12}) (q) \text{ and so } \epsilon_{\sigma, a_0} (\text{C}_{a_2}) = \epsilon_{\sigma, a_0} (\text{C}_{a_2'}) \cdot \epsilon_{\sigma_{\left| a_1 \right|}, q_{12}} (\text{C}_{a_2'}), \text{ which ends the induction and so the case}.

2. \( a_0 = a_{12} \).
   By definition, \( \epsilon_{\sigma, a_0} (\text{C}_{a_1}) = 1 \)
   a) Both \( a_{12} \notin a_0 \) and \( a_0 \notin a_{12} \). This implies \( a_{12} \notin a_0 \) and \( a_0 \notin a_{12} \). Then, by definition, \( \epsilon_{\sigma, a_0} (\text{C}_{a_{12}}) = \epsilon_{\sigma_{\left| a_1 \right|}, a_0} (\text{C}_{a_{12}}) = 0 \).

b) \( a_{12} \leq a_0 \). This implies \( a_{12} \leq a_{12}' \). Then, by definition, \( \epsilon_{\sigma, a_0} (\text{C}_{a_{12}}) = \epsilon_{\sigma_{\left| a_1 \right|}, a_0} (\text{C}_{a_{12}}) = 1 \).

c) \( a_0 \leq a_{12} \).
   We proceed by induction on size \( s \) of \( a_{12} \).
   Basis: \( s = 0 \), i.e. \( a_2 = q_{12} \). Then by definition \( \epsilon_{\sigma, a_0} (\text{C}_{a_{12}}) = \epsilon_{\sigma, a_0} (\text{C}_{a_1}) = 1 \). Moreover \( q_{12} \leq a_{12}' \) which means \( \epsilon_{\sigma_{\left| a_1 \right|}, a_0} (\text{C}_{a_{12}}) = 1 \), which ends the basis.
induction.
We assume the result to be true up to size $s \in \mathbb{N}$ and we want to show that it is still true for size $s + 1$. Let $a_2 \in Frags^+(\{A\}, f state(a_2) = lstate(a_1) \equiv q_1$ with $|a_2| = s + 1$. We note $a_2 = a_2^{-\sim} q' a q$ and $a_1' = a_1^{-\sim} a_2'$. We have $|a_2'| = s$ and $a_2' \leq a_1'$. By definition we have $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = \epsilon_{\sigma(a_1), a_2'}(C_{a_1}^{-\sim}) \cdot \sigma(a_2') \eta_1(A, q', a) \cdot \eta_1(A, q', a)(q')$.

In parallel, by definition: $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = \epsilon_{\sigma(a_1), a_2'}(C_{a_1}^{-\sim}) \cdot \sigma(a_1') \eta_1(A, q', a) \cdot \eta_1(A, q', a)(q)$ and by induction assumption, $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = \epsilon_{\sigma(a_1), a_2'}(C_{a_1}^{-\sim}) \cdot \sigma(a_2') \eta_1(A, q', a) \cdot \eta_1(A, q', a)(q)$ and so $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = \epsilon_{\sigma(a_1), a_2'}(C_{a_1}^{-\sim}) \cdot \sigma(a_2') \eta_1(A, q', a)$.

Finally, since $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = 1$, we have $\epsilon_{\sigma(a_1), a_2'}(C_{a_1}) = \epsilon_{\sigma(a_1), a_2'}(C_{a_1})$ which ends the induction, the case and so the proof.

We have formally defined our notion of creation-oblivious scheduler. This will be a key property to ensure lemma 11.23 that allows to reduce the measure of a class of comportment as a function of measures of classes of shorter comportment where no creation of $\mathcal{A}$ or $\mathcal{B}$ occurs excepting potentially at very last action. This reduction is more or less necessary to obtain monotonicity of implementation relation.

11.2 Creation made explicit

In this subsection, we recall notion of creation-explicitness (already introduced in subsection 6.2). This property will allow us to obtain the reduction of lemma 11.23 mentioned in last paragraph.

Definition 11.6 (creation-explicit PCA). Let $\mathcal{A}$ be a PSIOA and $X$ be a PCA. We say that $X$ is $\mathcal{A}$-creation-explicit iff: it exists a set of actions, noted creation-actions($X$), $\mathcal{A}$, s. t. $\forall q_X \in states(X)$, $\forall a \in \sim sig(X)(q_X)$, if we note $A_X = auth(config(X)(q_X))$ and $q_\mathcal{X} = created(X)(q_X)(a)$, then $A \notin A_X \land A \in q_\mathcal{X} \implies a \in creation-actions(X)(\mathcal{A})$.

This property of creation-explicitness will clarify the condition to obtain surjectivity of $\mu^{\mathcal{A}+}_e$ since it suffices to consider this function with a restricted range where no action of creation-actions($X$)($\mathcal{A}$) appears before last action.

Lemma 11.7 (Partial surjectivity with explicit creation). Let $\mathcal{A}$ be a PSIOA and $X$ be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA. Let $\tilde{E}$ be partially-compatible with $X$. Let $Y = X \setminus \mathcal{A}$. Let $E_{\mathcal{A}} = \tilde{E}|Y$. Let $((\tilde{E})||X)_{\mu^\mathcal{A}_e}$, $(\tilde{E})||X)_{\tilde{\mu}^\mathcal{A}_e}$, $(\tilde{E})||X)_{\mu_e^{\mathcal{A},+}}$, $(\tilde{E})||X)_{\tilde{\mu}^\mathcal{A}_e^{\mathcal{A},+}}$ the $\tilde{E}$-extension of $(X, \mu^\mathcal{A}_e, \tilde{\mu}^\mathcal{A}_e)$, $(X, \tilde{\mu}^\mathcal{A}_e^{\mathcal{A},+})$. Let $a, a' \in Exocs(E_{\mathcal{A}}||X)^{\mathcal{A}^{\sim}}$ s. t. creation-actions($X$)(\mathcal{A}) \cap actions(a) = \emptyset$

1) Then $\exists a \in dom(\mu^\mathcal{A}_e)$ s. t. $\mu^\mathcal{A}_e(a) = a \in \mathcal{A}$

2) If $a' = a^{-\sim} q, a, q' \in \mathcal{A}$ is creation-actions($X$)($\mathcal{A}$), then $\exists a \in dom(\tilde{\mu}^\mathcal{A}_e)$ s. t. $\mu^\mathcal{A}_e(a) = a'$

Proof. We proof the results in the same order they are stated in the lemma:

1) We note $\alpha = q_0, a_1, ..., a_n, q_n$... and we proof the result by induction on the prefix size $s$. Basis: the result trivially holds for any execution $a$ of size 0 by construction of $X \setminus \mathcal{A}$ that requires $X, \mu^\mathcal{A}_e(\text{start}(X)) = \text{start}(X \setminus \mathcal{A})$. We assume the result holds up to prefix size $s$ and we show it still holds for prefix size $s + 1$. We note $\alpha_s = q_0, a_1, ..., a_s, q_s$ and $\tilde{\alpha}^{\mathcal{A}+} \in Exocs(\tilde{E}|X)$ s. t. $\tilde{\mu}^\mathcal{A}_e(\tilde{\alpha}_s) = \alpha_s$. By lemma 9.17 of signature preservation $a^{s+1} \in sig(\tilde{E}|X)(\tilde{q}_s)$. Moreover, by assumption $a^{s+1} \notin \text{creation-actions}(X)(\mathcal{A})$ which means the application of lemma 9.8 of homomorphic transitions lead us to $\eta(\tilde{E}|X, \tilde{\alpha}^{\mathcal{A}+}, \tilde{\alpha}^{\mathcal{A}+}) = \tilde{\mu}^\mathcal{A}_e(\tilde{q}_s) = q_s$. So it exists $\tilde{\alpha}^{s+1} \in supp(\eta(\tilde{E}|X, \tilde{\alpha}^{\mathcal{A}+}, \tilde{\alpha}^{\mathcal{A}+}))$ with $\tilde{\mu}^\mathcal{A}_e(\tilde{\alpha}^{s+1}) = q$. This ends the induction and so the proof of 1.
(2) We apply 1. and note $\tilde{a} \in Exe\subset (\tilde{E})|X)$ s. t. $\tilde{\mu}_{\tilde{A}}^+(\tilde{a}) = a$. By lemma 9.17 of signature preservation $a \in sig(\tilde{E})|X(q)$ with $\tilde{q} = Istate(\alpha)$. Moreover, by lemma 9.8 of homomorphic transition, $\eta(\tilde{E})|X(q,a) \rightarrow \eta(\tilde{E})|X(q,a')$. So it exists $\tilde{q}' \in supp(\eta(\tilde{E})|X(q,a))$ with $\mu_{\tilde{A}}^+(\tilde{q}') = q'$. So $\mu_{\tilde{A}}^+(\tilde{a} \sim \tilde{q} \sim \tilde{q}') = \alpha'$ which ends the proof.

□

Since we i) classify executions in some classes according to their projection on an environment and ii) are concerned by the actions of the execution that create $\mathcal{A}$, the next lemma will simplify this classification. It states that if the projection $e$ of an execution $\alpha \in Exe\subset (\tilde{E})|\tilde{A}^w)$ on the environment $\tilde{E}, \mathcal{A}$ ends by an action $a \in creation-actions(X)(\mathcal{A})$, then the execution necessarily ends by $a$.

Lemma 11.8 (Environment projection ends on creation implies the execution itself ends on creation).

Let $\mathcal{A}$ be a PSIOA and $X$ be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA. Let $\tilde{E}$ be partially-compatible with $X$. Let $Y = X \setminus \{\mathcal{A}\}$. Let $\tilde{E}, \mathcal{A} = \tilde{E}|Y$.

Let $(((\tilde{E})|X, \tilde{\mu}_z^+, (\tilde{E})|X, \tilde{\mu}_z^+), ((\tilde{E})|X, \tilde{\mu}_r^+, (\tilde{E})|X, \tilde{\mu}_r^+))$ the $\tilde{E}$-extension of

$((X, \mu_z^+, X, \mu_z^+, X, \mu_z^+, X, \mu_z^+))$.

Let $\alpha \in creation-actions(X)(\mathcal{A})$ and $\alpha \in Exe\subset (\tilde{E})|\tilde{A}^w)$ s. t. $\alpha \uparrow \mathcal{A}, \mathcal{A} = e' = e \sim q, a, q'$ with creation-actions($X)(\mathcal{A}) \cap actions(e) = 0$.

Then it exist $\tilde{a} \in dom(\mu_{\tilde{A}}^+) s. t. \tilde{\mu}_{\tilde{A}}^+(\tilde{a}) = a$.

Proof. We note $\alpha = a^1 \sim q^1, a^2, q^2 \sim a^2$.

We have $q^1 \uparrow \tilde{A}^w = q^1, a^2$ with $\mathcal{A}$, $q^1 \uparrow \tilde{A}^w = q^1, a^2$. Indeed let assume the contrary: $q^1 \uparrow \tilde{A}^w = q^1, a^2$. Then $q \uparrow \tilde{A}^w = q^1, a^2$ for every state $q \in a^1$. Since creation-actions($X)(\mathcal{A}) \cap actions(e) = 0$, creation-actions($X)(\mathcal{A}) \cap actions(a^1) = 0$. Thus we apply lemma 11.7 of partial surjectivity with explicit creation to obtain, it exists $\tilde{a}^1 \in Exe\subset (\tilde{E})|X$ s. t. $\tilde{\mu}_{\tilde{A}}^+(\tilde{a}^1) = a^1$ with both $\mathcal{A} \in aux(config(X)(Istate(\tilde{a}^1) \uparrow X))$ and $\tilde{a} \in creation-actions(X)(\mathcal{A}) \cap sig(X)(Istate(\tilde{a}^1) \uparrow X)$ which is impossible.

Since $q^1 \uparrow \tilde{A}^w = q^1, a^2, q \uparrow \tilde{A}^w = q^1, a^2$ for every state $q \in a^2$. Hence, $a^2 = q^1, a^2$ to respect $a \uparrow \mathcal{E}, \mathcal{A} = e'$, which means $\alpha = a^1 \sim q^1, a^2, a^2 \sim q^2$.

Since creation-actions($X)(\mathcal{A}) \cap actions(e) = 0$, creation-actions($X)(\mathcal{A}) \cap actions(a^2) = 0$. Thus we apply lemma 11.7 of partial surjectivity with explicit creation to obtain $\exists \tilde{a} \in Exe\subset (\tilde{E})|X$ s. t. $\tilde{\mu}_{\tilde{A}}^+(\tilde{a}) = a$.

□

Here we recall the notion of print (already introduced in 4.6). This notion captures the perception of a system $\tilde{E}|X$ by the environment $\tilde{E}$. This notion allows us to propose an intuitive definition of implementation that is monotonic w. r. t. PSIOA creation.

Definition 11.9 (print = trace + environment projection). Let $K$ be a PSIOA (resp. a PCA). Let $\tilde{E}$ be a PSIOA (resp. a PCA) partially-compatible with $K$. We note $print_{(E,K)} \alpha \in Exe\subset (\tilde{E})|K \rightarrow (\text{trace}_{\tilde{E}})|K(\alpha), \alpha \uparrow \tilde{E}$). For every $\zeta = (\tilde{E}, e)$ where $\tilde{E}$ is a sequence of actions and $e$ an alternating sequence of states and action, we note $Class(Exe\subset (\tilde{E})|K), print_{(E,K)} \zeta) = \{\alpha \in Exe\subset (\tilde{E})|K)|print_{(E,K)} \alpha = \zeta\}.$

The next definition allows to aggregate print-based classification into a larger classification in order to prepare the reduction of the implementation of a system $X_K$ by a system $X, \mathcal{A}$ into the implementation of $B$ by $\mathcal{A}$.
Definition 11.10 (aggregate print without creation). Let \( A \) be a PSIOA, let \( X \) be a \( A \)-conservative and \( A \)-creation-explicit PCA. Let \( E \) be a PCA partially-compatible with \( X \). Let \( Y = X \setminus \{ A \} \). For every \( \zeta = (b, e) \) where \( b \) is a sequence of actions and \( e = e^\sim q' a q \) an alternating sequence of states and actions with actions(charge) \( \cap \) creation-actions(X)\( (\mathcal{A}) = 0 \), then we note

\[
\text{Class}(\text{Execs}(\tilde{E}||X), \text{print}_{\text{proxy}, A}(\tilde{E}), \zeta) = \{ \tilde{a} \in \text{Execs}(\tilde{E}||X) | \text{trace}(\tilde{a}) = b \land (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - (\tilde{a}) \uparrow (\tilde{E}||Y) = e \}.
\]

Now we state the bijectivity of \( \dot{\mu}_e^\mathcal{A} - + \) (when no creation occurs) in terms of corresponding classes of external perception.

Lemma 11.11 (\( \dot{\mu}_e^\mathcal{A} - + \) is a bijection from \( \tilde{C} \) to \( C \)). Let \( A \) be a PSIOA and \( X \) be a \( A \)-conservative and \( A \)-creation-explicit PCA. Let \( \tilde{E} \) be partially-compatible with \( X \). Let \( Y = X \setminus \{ A \} \). Let \( E,A = \tilde{E}||Y \). Let \( ((\tilde{E}||X), \dot{\mu}_e^\mathcal{A}, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +), (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +) \) the \( \tilde{E} \)-extension of \((X, \dot{\mu}_e^\mathcal{A}, X, \dot{\mu}_e^\mathcal{A} - +), X, \dot{\mu}_e^\mathcal{A} - +, X, \dot{\mu}_e^\mathcal{A} - +) \).

For every \( \zeta = (b, e = e^\sim q, a, q') \) where \( b \) is a sequence of actions and \( e' \) an alternating sequence of states and action with creation-actions(X)\( (\mathcal{A}) \cap \) actions(e) = 0, \( (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - + \) is a bijection from \( \tilde{C} \) to \( C \), where

\[
\begin{align*}
\tilde{C} &= \text{Class}(\text{Execs}(\tilde{E}||X), \text{print}_{\text{proxy}, A}(\tilde{E},X), \zeta) \\
C &= \text{Class}(\text{Execs}(\tilde{E}||X), \text{print}_{\text{proxy}, A}(\tilde{E},X), \zeta)
\end{align*}
\]

Proof. 

\( \bullet \) Injectivity is immediate by lemma 7.11, item (2).

\( \bullet \) Surjectivity: Let \( \alpha \in C \). By previous lemma 11.7, \( \exists \tilde{a} \in \text{Execs}(\tilde{E}||X) \) s. t. \( \dot{\mu}_e^\mathcal{A} - + (\tilde{a}) = \alpha \). Finally, since \( \dot{\mu}_e^\mathcal{A} - + \) is a strong executions-matching, the trace is preserved which means \( \text{trace}(\tilde{a}) = \text{trace}(\alpha) \).

\[ \square \]

This bijectivity allows us to obtain the preservation of measure of probability for corresponding classes of external perception.

Lemma 11.12 (equivalence of top/distribution corresponding cones). Let \( A \) be a PSIOA and \( X \) be a \( A \)-conservative and \( A \)-creation-explicit PCA. Let \( \tilde{E} \) be partially-compatible with \( X \). Let \( Y = X \setminus \{ A \} \). Let \( E,A = \tilde{E}||Y \). Let \( ((\tilde{E}||X), \dot{\mu}_e^\mathcal{A}, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +), (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +) \) the \( \tilde{E} \)-extension of \((X, \dot{\mu}_e^\mathcal{A}, X, \dot{\mu}_e^\mathcal{A} - +), X, \dot{\mu}_e^\mathcal{A} - +, X, \dot{\mu}_e^\mathcal{A} - +) \).

Let \( \zeta = (b, e' = e^\sim q, a, q') \) where \( b \) is a sequence of actions and \( e' \) an alternating sequence of states and action with creation-actions(X)\( (\mathcal{A}) \cap \) actions(e) = 0.

\[
\begin{align*}
\tilde{C} &= \bigcup_{\alpha \in C} \{ \tilde{a}' \in \text{Execs}(\tilde{E}||X) | \tilde{a}' \preceq \tilde{a}' \} \\
C &= \bigcup_{\alpha' \in C} \{ \alpha' \in \text{Execs}(\tilde{E}||X) | \alpha \preceq \alpha' \}
\end{align*}
\]

Then for every \( \tilde{a} \in \text{executors}(\tilde{E}||X) \), for \( \sigma \) \( (((\tilde{E}||X), \dot{\mu}_e^\mathcal{A}, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +), (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +, (\tilde{E}||X), \dot{\mu}_e^\mathcal{A} - +) \)-alter ego of \( \tilde{a} \),

\[
\epsilon_{\tilde{a}, \delta_{\tilde{a}}}((\tilde{E}||X), (\tilde{C}) = \epsilon_{\sigma, \delta_{\tilde{a}}}((\tilde{E}||X), (\tilde{C})
\]

Proof. By lemma 11.11, \( \dot{\mu}_e^\mathcal{A} - + \) is a bijection from \( \tilde{C} \) to \( C \). We note \( \{ (\tilde{a}_i, \alpha_i) \}_i \in \tilde{C} \times C \) the related pair of executions s. t. \( \dot{\mu}_e^\mathcal{A} - + (\tilde{a}_i) = \alpha_i \). We obtain \( \epsilon_{\tilde{a}_i, \delta_{\tilde{a}}((\tilde{E}||X), (\tilde{C}) = \sum_{i \in I} \epsilon_{\tilde{a}_i, \delta_{\tilde{a}}((\tilde{E}||X), (\tilde{C}) \) and \( \epsilon_{\sigma, \delta_{\tilde{a}}}((\tilde{E}||X), (\tilde{C}) = \sum_{i \in I} \epsilon_{\sigma, \delta_{\tilde{a}}}((\tilde{E}||X), (\tilde{C}) \).

Thus it is enough to show that \( \forall i \in I \), \( \epsilon_{\tilde{a}_i, \delta_{\tilde{a}}((\tilde{E}||X), (\tilde{C}) \) which is given by theorem 7.10 that can be applied since \( \dot{\mu}_e^\mathcal{A} - + \) is a continued executions-matching by theorem 9.23.

\[ \square \]

In next subsection, we want to extend this lemma 11.12 to any case with an arbitrary number of destructions and creations of the sub-automaton \( A \).

88
11.3 Monotonicity

In this subsection, we want to reduce the computation of the probability measure of a class of executions to the computation of several classes without creation (excepting at very last action) in order to apply lemma 11.12 and then use the implementation of $\mathcal{B}$ by $\mathcal{A}$.

We start by a sequence of definitions to identify easily an execution $\alpha$ as the concatenations of several execution fragments $\alpha_i$ where each $\alpha_i$ does not contain creation actions except for very last action.

**Definition 11.13 (n-building-vector for traces).** Let $\beta$ be a sequence of actions. Let $n \in \mathbb{N} \cup \{\infty\}$. A $n$-building-vector of $\beta$ is a (potentially infinite) vector $\vec{\beta} = (\beta^1, ..., \beta^t, ...) \text{ of } |\vec{\beta}| = n \text{ sequences of actions s. t. } \beta^1 \ldots \beta^t \ldots \beta^i \ldots = \beta$. We note Building-vectors($\beta$, $n$) the set of $n$-building-vector of $\beta$ and $\vec{\beta} \xrightarrow{n} \beta$ to say $\vec{\beta} \in$ Building-vectors($\beta$, $n$). We note $\vec{\beta}[i] = \beta^i$ and $\vec{\beta}[i] = \beta^1 \ldots \beta^i$.

**Definition 11.14 (n-building-vector for executions).** Let $e$ be an alternating sequence states and actions starting by state and finishing by a state if $e$ is finite. Let $n \in \mathbb{N} \cup \{\infty\}$. A $n$-building-vector of $e$ is a (potentially infinite) vector $\vec{e} = (e^1, ..., e^s, ...) \text{ of } |\vec{e}| = n \text{ alternating sequences of states and actions starting by state and finishing by a state (excepting potentially the last one if it is infinite) s. t. } e^1 \ldots e^s \ldots e^i \ldots = e \text{ (with } \forall i \in [1, |e| - 1], fstate(e_{i+1}) = lstate(e_i)).$

We note Building-vectors($e$, $n$) the set of $n$-building-vector of $e$ and $\vec{e} \xrightarrow{n} e$ to say $\vec{e} \in$ Building-vectors($e$, $n$). We note Building-vectors($e$) = $\bigcup_{n \in \mathbb{N} \cup \{\infty\}}$ Building-vectors($e$, $n$) and $\vec{e} : e$ to say $e \in$ Building-vectors($e$). We note $\vec{e}[i] = e^i$ and $\vec{e}[i] = e^1 \ldots e^i$.

**Definition 11.15 (e : (X, $\mathcal{A}$) e).** Let $W$ and $X$ be two PCA s. t. $X$ is $\mathcal{A}$-creation-explicit, $e \in Frags(W)$. We note $\vec{e} : e$ (and $\vec{e} : e$ when $X$ is clear in the context) the (clearly unique) vector $\vec{e} \in$ Building-vectors($e$) of execution fragments s. t.

1. $\forall i \in [1, n], \text{ (actions}(\vec{e}[i]) \setminus \text{laction}(\vec{e}[i])) \cap \text{creation-actions}(X)(\mathcal{A}) = \emptyset$ and
2. $\forall i \in [1, n - 1], \text{laction}(\vec{e}[i])) \in \text{creation-actions}(X)(\mathcal{A})$.

We write $\vec{e} : e \xrightarrow{n} (X, \mathcal{A})$ or $\vec{e} : e \xrightarrow{n} \mathcal{A}$ to indicate that $|\vec{e}| = n$.

**Definition 11.16 ($\mathcal{A}$-decomposition).** Let $\mathcal{A}$ be a PSIOA and $X$ be a PCA. Let $\alpha = q^0a^1...an^a... \in Frags(X)$. We say that

- $\alpha$ is a $\mathcal{A}$-open-portion iff $\alpha$ does not create $\mathcal{A}$, i. e. $\forall i \in [1, |\alpha|], \mathcal{A} \notin \text{austs}(\text{config}(X)(q^{i-1})) \implies \mathcal{A} \notin \text{austs}(\text{config}(X)(q^i))$.
- $\alpha$ is a $\mathcal{A}$-closed-portion iff $\alpha$ does not create $\mathcal{A}$ excepting at very last action, i. e. $\forall i \in [1, |\alpha|], \mathcal{A} \notin \text{austs}(\text{config}(X)(q^{i-1})) \land \mathcal{A} \in \text{austs}(\text{config}(X)(q^i)) \implies i = |\alpha|$.
- $\alpha$ is a $\mathcal{A}$-portion of $X$ if it is either a $\mathcal{A}$-open-portion or a $\mathcal{A}$-closed-portion.

We call $\mathcal{A}$-decomposition of $\alpha$, note $\mathcal{A}$-decomposition($\alpha$), the unique vector $(\alpha^1, ..., \alpha^n, ...) \in$ Building-vectors($\alpha$) s. t.

- $\forall i \in [1, |\mathcal{A}$-decomposition($\alpha$)| - 1], $\alpha^i$ is a $\mathcal{A}$-closed-portion of $X$ and
- if $|\mathcal{A}$-decomposition($\alpha$)| = $n \in \mathbb{N}$, $\alpha^i$ is a $\mathcal{A}$-portion of $X$.

**Lemma 11.17 ($\vec{\alpha} : (X, \mathcal{A}) \alpha$ means $\vec{\alpha} = \mathcal{A}$-decomposition($\alpha$)).** Let $\mathcal{A}$ be a PSIOA and $X$ be a $\mathcal{A}$-creation-explicit PCA.

Let $\alpha \in Frags(X)$. Let $\vec{\alpha} = \mathcal{A}$-decomposition($\alpha$). Then $\vec{\alpha} : (X, \mathcal{A}) \alpha$.  

89
Proof. By definition, $\vec{a} \in \text{Building-vectors}(a)$. Still by definition, $\forall i \in [1, |A| - 1]$, $a^i$ is a $A$-closed-portion of $X$, i.e. $a^i$ does not create $A$ excepting at very last last action $\text{laction}(a_i)$. By definition of creation-explicitness, the two item of definition 11.15 are verified for every $i \in [1, |A| - 1]$. Finally, by definition, if $|A| = n \in \mathbb{N}$, $a^n$ is a $A$-portion of $X$, i.e. $a^n$ does not create $A$ excepting potentially at very last last action if $a^n$ is finite. Again, by definition of creation-explicitness, the first item of definition 11.15 is verified.

After this sequence of definitions we can start our process of decomposition. First, we partition each class of equivalence into an aggregation several "proxy classes" that will able the application of lemma 11.12 after their decomposition.

**Lemma 11.18 (Partitioning of class $\tilde{C}^{\vec{a}, \vec{e}}$ into classes $\tilde{C}^{\vec{a}, \vec{e}}$).** Let $\mathcal{A}$ be a PSIQA, let $X$ be a $A$-conservative PCA and $\tilde{E}$ partially-compatible with $X$. Let $\vec{b}$ be a sequence of actions, let $\tilde{e}$ an alternating sequence of states and actions starting by a state and finishing by a state if finite and let $\vec{z} = (\vec{b}, \vec{e})$. Let

- $\tilde{C}^{\vec{z}} \triangleq \text{Class(Execs}(\tilde{E}||X), \text{print}_X(\tilde{E}||X), \tilde{e}) \triangleq \{\tilde{a} \in \text{Execs}(\tilde{E}||X) | \text{print}_X(\tilde{E}||X)(\tilde{a}) = \tilde{e}\}$.
- $\tilde{E} = \{\tilde{C}^{\vec{a}, \vec{e}} | n \in \mathbb{N} \cup \{\infty\}, \vec{b} \in \in \vec{e} \in \text{Execs}(\tilde{E}||X \setminus \{A\}), e \uparrow \tilde{E} = \tilde{e}, \vec{e} = \tilde{e}\} \text{, where}$
- $\tilde{C}^{\vec{a}, \vec{e}} = \{(a, a) = \text{A-decomposition}(a), \forall i \in [1, |a|]$, $\text{trace}(a[i]) = \beta[i], (\tilde{E}||X).\mu^{\vec{a}}(a[i]) \uparrow (\tilde{E}||X \setminus \{A\}) = \tilde{e}[i]\}$

Then $\tilde{E}$ is a partition of $\tilde{C}^{\vec{z}}$.

Proof. The proof is immediate by construction since the $A$-decomposition is unique. We first show that $\tilde{C}^{\vec{b}, \vec{e}} = \bigcup_{n \in \mathbb{N} \cup \{\infty\}} \bigcup_{\tilde{E} = \tilde{a}} \bigcup_{\vec{e} \in \text{Execs}(\tilde{E}||X \setminus \{A\})} \tilde{C}^{\vec{a}, \vec{e}}.$

By double inclusion:

- $\subseteq \{\tilde{a} \in \text{A-decomposition}(a)$. We note $\overrightarrow{a} = \text{A-decomposition}(a)$. We note $\overrightarrow{b} = \text{trace}(\overrightarrow{a}), \overrightarrow{e} = \text{trace}(\overrightarrow{a}) \uparrow \tilde{E}, \overrightarrow{e} = (\tilde{E}||X).\mu^{\vec{a}}(\tilde{e})$. By construction, we have $\overrightarrow{b} \overset{\sim}{=} \overrightarrow{b}$ and $\overrightarrow{e} = \overrightarrow{e}$ for some $n \in \mathbb{N} \cup \{\infty\}$. Thus $\overrightarrow{a} \in \bigcup_{n \in \mathbb{N} \cup \{\infty\}} \bigcup_{\tilde{E} = \tilde{a}} \bigcup_{\vec{e} \in \text{Execs}(\tilde{E}||X \setminus \{A\})} \tilde{C}^{\vec{a}, \vec{e}}$

Hence $\tilde{C}^{\vec{b}, \vec{e}} \subseteq \bigcup_{n \in \mathbb{N} \cup \{\infty\}} \bigcup_{\tilde{E} = \tilde{a}} \bigcup_{\vec{e} \in \text{Execs}(\tilde{E}||X \setminus \{A\})} \tilde{C}^{\vec{a}, \vec{e}}$. By construction, $\text{trace}(a) = \beta$ and $a \uparrow \tilde{E} = \tilde{e}$.

Hence, $\bigcup_{n \in \mathbb{N} \cup \{\infty\}} \bigcup_{\tilde{E} = \tilde{a}} \bigcup_{\vec{e} \in \text{Execs}(\tilde{E}||X \setminus \{A\})} \tilde{C}^{\vec{a}, \vec{e}} = \tilde{C}^{\vec{b}, \vec{e}}$. We show that $\forall i((\overrightarrow{b}, \overrightarrow{e}), (\overrightarrow{b}', \overrightarrow{e}'))$ s.t. $(\overrightarrow{b}, \overrightarrow{e}) \neq (\overrightarrow{b}', \overrightarrow{e}')$, $\tilde{C}^{\overrightarrow{b}, \overrightarrow{e}} \cap \tilde{C}^{\overrightarrow{b}', \overrightarrow{e}'} = \emptyset$. Let $(a, a') \in \tilde{C}^{\overrightarrow{b}, \overrightarrow{e}} \times \tilde{C}^{\overrightarrow{b}', \overrightarrow{e}'}$ and $\overrightarrow{a} = \text{A-decomposition}(a)$ and $\overrightarrow{a}' = \text{A-decomposition}(a')$. If $\overrightarrow{a} \neq \overrightarrow{a}'$, then necessarily $a \neq a'$. We proceed by contradiction. Let assume $\overrightarrow{a} = \overrightarrow{a}'$. We note $n = |a| = |a'|$ with $n \in \mathbb{N} \cup \{\infty\}$. Then $\forall i \in [1, n]$, $\text{trace}(a[i]) = \beta[i] = \beta'[i]$ and $(\tilde{E}||X).\mu^{\vec{a}}(a[i]) \uparrow \tilde{E} = \tilde{e}[i] = \overrightarrow{e}[i]$ which is not possible since $(\overrightarrow{b}, \overrightarrow{e}) \neq (\overrightarrow{b}', \overrightarrow{e}')$. Thus $\overrightarrow{a} \neq \overrightarrow{a}'$ which means $a \neq a'$ and so $\forall i((\overrightarrow{b}, \overrightarrow{e}), (\overrightarrow{b}', \overrightarrow{e}'))$ s.t. $(\overrightarrow{b}, \overrightarrow{e}) \neq (\overrightarrow{b}', \overrightarrow{e}')$, $\tilde{C}^{\overrightarrow{b}, \overrightarrow{e}} \cap \tilde{C}^{\overrightarrow{b}', \overrightarrow{e}'} = \emptyset$. 

\[]
Now we use the $\mathcal{A}$-creation explicitness to prove the lemma 11.19 and 11.20 in order to restate the previous lemma 11.18 into a simpler version to use, stated in lemma 11.21.

**Lemma 11.19 (Chunks ending on creation).** Let $\mathcal{A}$ be a PSIOA, let $X$ be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\hat{E}$ partially-compatible with $X$. Let $\tilde{a} \in \text{Frags}(\hat{E}||X)$ and $e \in \text{Frags}(\hat{E}||X \setminus \{\mathcal{A}\})$ s.t. $(\hat{E}||X,\mu_{\mathcal{A}^+}^\mathcal{E})(\tilde{a}) \uparrow (\hat{E}||X \setminus \{\mathcal{A}\}) = e$.

Then
- $\text{laction}(\tilde{a}) = a_i \in \text{creation-actions}(X)(\mathcal{A}) \implies \text{laction}(e) = a_i \in \text{creation-actions}(X)(\mathcal{A})$.
- If $\tilde{a} \in \text{dom}(\mu_{\mathcal{A}^+}^\mathcal{E})$, then $\text{laction}(\tilde{a}) = a_i \in \text{creation-actions}(X)(\mathcal{A}) \iff \text{laction}(e) = a_i \in \text{creation-actions}(X)(\mathcal{A})$.

**Proof.** We prove the two implications in the same order.

- $\implies$ Let assume $a_i \subseteq \text{laction}(\tilde{a}) \subseteq \text{creation-actions}(X)(\mathcal{A})$. Since $X$ is $\mathcal{A}$-creation-explicit, we have $\tilde{a} = \tilde{a}' \gamma a_0 q_0$ with $\mathcal{A} \not\subseteq \text{auts}(\text{config}(X)(q_0))$. Thus $\text{laction}(e) = a_i \in \text{creation-actions}(X)(\mathcal{A})$.

- $\Leftarrow$ Let assume $a_i \subseteq \text{creation-actions}(\tilde{a}) \subseteq \text{creation-actions}(X)(\mathcal{A})$. Thus $a_i \in \text{actions}(\tilde{a})$. Since $X$ is $\mathcal{A}$-creation-explicit, it implies $\tilde{a} = \tilde{a}' \gamma a_0, q_0 \gamma \tilde{a} \gamma$ where $\mathcal{A} \not\subseteq \text{auts}(\text{config}(X)(q_0))$ and $\mathcal{A} \subseteq \text{auts}(\text{config}(X)(q_0))$. But $\tilde{a} \in \text{dom}(\hat{E}||X,\mu_{\mathcal{A}^+}^\mathcal{E})$, so $\tilde{a}' = \tilde{a}'_0$ and hence $\text{laction}(\tilde{a}) = a_i \in \text{creation-actions}(X)(\mathcal{A})$.

**Lemma 11.20 (Common states for pasting).** Let $\mathcal{A}$ be a PSIOA, let $X$ be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\hat{E}$ partially-compatible with $X$. Let $\beta$ be a sequence of actions, let $e$ be an alternating sequence of states and actions starting by a state and finishing by a state. Let $n \in \mathbb{N} \cup \{\infty\}$, $\beta \gamma^n \beta$, $\gamma \in \mathcal{E}$. Let $\mathcal{C}^{\mathcal{E}} \beta = \{\alpha[i] = \alpha = \text{A-decomposition}(\alpha), \forall i \in [1,|\mathcal{C}|], \text{trace}(\alpha[i]) = \beta[i], (\hat{E}||X,\mu_{\mathcal{A}}(\alpha[i]) \uparrow (\hat{E}||X \setminus \{\mathcal{A}\}) = \gamma[i]$.\)

1. If $\mathcal{C}^{\mathcal{E}} \beta \neq \emptyset$, then
   - (a) $e = (X,\mathcal{A}) \in \mathcal{C}^{\mathcal{E}} \beta$  
   - (b) $\forall \alpha, \alpha' \subseteq \mathcal{C}^{\mathcal{E}} \beta$, for $\alpha = \alpha[i]$ and $\alpha' = \alpha'[i]$, $\forall i \in [1,|\alpha| - 1], \text{lstate}(\alpha[i]) = \text{lstate}(\alpha'[i]) = \text{fstate}(\alpha[i+1]) = \text{fstate}(\alpha'[i+1])$.

2. If $e \neq \mathcal{C}^{\mathcal{E}} \beta$, then $\mathcal{C}^{\mathcal{E}} \beta = \emptyset$.

**Proof:**

1. Let $\alpha \in \mathcal{C}^{\mathcal{E}} \beta$ and $\alpha = \text{A-decomposition}(\alpha)$. Since $X$ is $\mathcal{A}$-creation-explicit, we have $\alpha[i]$ that ends on creation of $\mathcal{A}$ and $(\hat{E}||X,\mu_{\mathcal{A}}(\alpha[i]) \uparrow (\hat{E}||X \setminus \{\mathcal{A}\}) = \gamma[i]$. Thus, since $X$ is $\mathcal{A}$-creation-explicit, we can apply previous lemma 11.19 to obtain $\forall i \in [1,|\alpha| - 1], \text{laction}(\alpha[i]) \subseteq \text{creation-actions}(X)(\mathcal{A})$. Let $i \in \mathcal{V}$ and $\gamma'$ a strict prefix of $\gamma[i]$. By construction of $\mu_{\mathcal{A}^+}^\mathcal{E}$, it exists a strict prefix of $\alpha[i]$, noted $\alpha'[i]$, s.t. $(\hat{E}||X,\mu_{\mathcal{A}}(\alpha'[i]) \uparrow (\hat{E}||X \setminus \{\mathcal{A}\}) = \gamma'[i]$. By definition of $\alpha[i]$, $\alpha = \text{A-decomposition}(\alpha)$, $\text{laction}(\alpha'[i]) \subseteq \text{creation-actions}(X)(\mathcal{A})$. By contraposition of lemma 11.19, $\text{laction}(\alpha'[i]) \subseteq \text{creation-actions}(X)(\mathcal{A})$. Since the results holds for any strict prefix $\gamma'[i]$.

2. Let $\alpha, \alpha' \subseteq \mathcal{C}^{\mathcal{E}} \beta$, Let $\alpha = \text{A-decomposition}(\alpha)$, $\alpha' = \text{A-decomposition}(\alpha')$, let $|\alpha| = |\alpha'| = n$ and let $i \in [1,|\alpha| - 1]$. By construction we have directly $\text{lstate}(\alpha[i]) = \text{fstate}(\alpha[i+1])$ and $\text{lstate}(\alpha'[i]) = \text{fstate}(\alpha'[i+1])$. 

91
fstate(α' | i + 1)). Moreover, we have both (1) (∃E||X).μE(A(α | i)) ⊢ (∃E||X \ {A}) = (∃E||X).μA(A(α | i)) ⊢ (∃E||X \ {A}) = e | i and (2) map(config(X)(lstate(α' | i)))(A) = map(config(X)(lstate(α' | i)))(A) = q. The property (1) implies (3) map(config(X)(lstate(α' | i)))(A) = map(config(X)(lstate(α' | i)))(A) = [[]{A}] and ∀B ∈ A, map(config(X)(lstate(α' | i)))(B) = map(config(X)(lstate(α' | i)))(B). Since X is configuration-conflict-free, lstate(α' | i)) = lstate(α' | i)) which ends the proof.

(2) By contraposition of former item 1a.

□

Now we can simplify lemma 11.17. We can already guess in \( \hat{C}^{e, \beta} \) the form of "proxy aggregated class" introduced in definition 11.10.

Lemma 11.21 (Partitioning of class \( \hat{C}^{e, \beta} \) into classes \( \hat{C}^{e, \beta} \) ending on \( X \) creation). Let \( \mathcal{A} \) be a PSAO, let \( X \) be a \( \mathcal{A} \)-conservative and \( \mathcal{A} \)-creation-explicit PCA and \( \hat{E} \) partially-compatible with \( X \). Let \( E, \mathcal{A} = \hat{E}|(X \setminus \{ A \}) \). Let \( \beta \) be a sequence of actions, let \( \hat{e} \) an alternating sequence of states and actions starting by a state and finishing by a state if finite and let \( \hat{\xi} = (\beta, \hat{e}) \). Let

- \( \hat{C}^{\hat{\xi}} = \text{Class}(\text{Execs}(\hat{E}||X), print(\hat{E}||X), \hat{\xi}) \). Let
- \( \hat{C}^{\hat{\xi}, \mathcal{E}} = \hat{C}^{\hat{\xi}} \exists \xi \in \text{Execs}(E, \mathcal{A}), e \vdash \hat{E} = \hat{e}, e \vdash \mathcal{E} = \mathcal{E}_n \) where
- \( \hat{C}^{e, \beta} = \{ \alpha \in \text{Execs}(\hat{E}||X) | \alpha \vdash \hat{E}, \forall i \in [1, |\alpha|], \alpha[i] = e[i] \} \). Let
- \( \hat{C}^{e, \beta} = \{ \alpha \in \text{Execs}(\hat{E}||X) | \alpha \vdash \hat{E}, \forall i \in [1, |\alpha|], \alpha[i] = e[i] \} \).

Then, \( \hat{C}^{e, \beta} \) is a partition of \( \hat{C}^{\hat{\xi}} \).

Proof. By conjunction of lemma 11.17, lemma 11.18 and lemma 11.20, item 1a.

In next paragraph, we will isolate "proxy aggregated class without creation" introduced in definition 11.10.

Decomposition. We start this paragraph, by showing that \( \hat{C}^{e, \beta} \) is equal to the set of concatenated executions issued to some fixed "proxy aggregated classes without creation" introduced in definition 11.10.

Lemma 11.22 (Decomposition into simple classes). Let \( \mathcal{A} \) be a PSAO, \( X \) be a \( \mathcal{A} \)-conservative and \( \mathcal{A} \)-creation-explicit a PCA and \( \hat{E} \) partially-compatible with \( X \). Let \( E, \mathcal{A} = \hat{E}|(X \setminus \{ A \}) \). Let \( \beta \) be a sequence of actions, let \( e \in \text{Execs}(E, \mathcal{A}) \). Let \( n \in \mathbb{N} \cup \{ \infty \}, e = e^n \). Let

\( \hat{C}^{e, \beta} = \{ \alpha \in \text{Execs}(\hat{E}||X) | \alpha \vdash \hat{E}, \forall i \in [1, |\alpha|], \alpha[i] = e[i] \} \).

Then, \( \hat{C}^{e, \beta} \) with

- \( \hat{C}^{e, \beta} = \{ \alpha \in \text{Execs}(\hat{E}||X) | \alpha \vdash \hat{E}, \forall i \in [1, |\alpha|], \alpha[i] = e[i] \} \).
- \( \hat{E}^1 = \hat{E} \) and \( \forall i \in [2, n], \hat{E}^i = \hat{E}_{q_{\xi} = q_{\xi}} \) with \( q_{\xi} = fstate(e_i) \).

as per definition 9.9.
• $X^1 = X$ and $\forall i \in [2,n], X^i = X_{q^i_X \to q^i_X}$ (as per definition 9.9) with $q^i_X$ the unique state (by configuration-conflict-free property) s.t. $\text{config}(X)(q^i_X) = C^i_Y \cup C^i_A$ with $C^i_Y = \text{config}(X \setminus \{A\})(f\text{state}(e_i) \uparrow (\hat{E}||X \setminus \{A\}))$ and $C^i_A = (\{A\}, (A, q_A))$.

\[ \bigotimes_i C^i \subseteq C \otimes C^2 \otimes \ldots \otimes C^n \]

\[ C^1 \otimes C^2 = \{a_1 \otimes a_2 | a_1 \in C, a_2 \in C^2 \} \]

\[ \text{Proof. By double inclusion.} \]

• $\exists \alpha \in \hat{C}^+ \hat{\beta}$, i.e. $\alpha = \mathcal{A}$-decomposition($\alpha$) and $\forall i \in [1,|\alpha|], \text{trace}(\alpha[i]) = \hat{\beta}[i], (\hat{E}||X).\mu^A(\alpha[i]) \uparrow (\hat{E}||X \setminus \{A\}) = \alpha[i]$. We need to show that $\forall i \in [1,|\alpha|], \alpha[i] \in \hat{C}^+ \hat{\beta}[i]$. By construction due to $\mathcal{A}$-decomposition, $\forall i \in [1,|\alpha|], f\text{state}(\alpha[i]) = \text{state}(\alpha[i-1])$ where $\alpha[i-1]$ ends on $\mathcal{A}$-creation (1). Moreover, $\forall i \in [1,|\alpha|], (\hat{E}||X).\mu^A(\alpha[i]) \uparrow (\hat{E}||X \setminus \{A\}) = \alpha[i]$ (2). By (1) and (2), $f\text{state}(\alpha[i]) = \text{start}(\hat{E}^i||X^i)$ where $\hat{E}^i$ and $X^i$ are defined like in the lemma (3). By construction due to $\mathcal{A}$-decomposition, $\alpha[i]$ does not create $\mathcal{A}$ before its very last action (4), thus by (3) and (4) $\alpha[i] \in \text{dom}((\hat{E}^i||X^i).\mu^A(\mathcal{A}))$ (5). Then $\hat{C}^+ \hat{\beta} \subseteq \bigotimes_i C^+ \hat{\beta}[i]$. 

\[ \exists \alpha \in \hat{C}^+ \hat{\beta}, \text{ i.e. } \alpha = \mathcal{A}$-decomposition($\alpha$) and $\forall i \in [1,|\alpha|], \text{trace}(\alpha[i]) = \hat{\beta}[i], (\hat{E}||X).\mu^A(\alpha[i]) \uparrow (\hat{E}||X \setminus \{A\}) = \alpha[i]. \]

Now we want to transform the term $\epsilon_{\hat{\sigma}}(C^+ \hat{\beta}[i], \hat{\beta}[i])$ as a function of some terms $\epsilon_{\hat{\sigma}'}(C^+ \hat{\beta}[i], \hat{\beta}[i])$ where $\hat{\sigma}'$ must be defined. The critical point is that the occurrence of these events might not be independent with (*) a perfect-information scheduler that chooses the measure of class $\hat{C}^+ \hat{\beta}[i], \hat{\beta}[i]$ as a function of the concrete prefix in class $\hat{C}^+ \hat{\beta}[i-1], \hat{\beta}[i-1]$. This observation enforced us to weaken the implementation definition to make it monotonic w.r.t. PSIOA creation by handling only creation-oblivious schedulers that cannot make the choice (*).
LEMMA 11.24 (MEASURE AFTER DECOMPOSITION FOR OBVIOUS CREATION SCHEDULER). Let \( \mathcal{A} \) be a PSIOA, \( X \) be a \( \mathcal{A} \)-conservative, \( \mathcal{A} \)-creation-explicit PCA and \( \tilde{E} \) partially-compatible with \( X \).

Let \( \beta \) be a vector of sequences of actions, let \( e \in \text{Execs}(\tilde{E} || X \setminus \{ \mathcal{A} \}) \), and \( e \xrightarrow{(\mathcal{X},\mathcal{A})} \). Let \( \delta \in \text{schedulers}(\tilde{E} || X) \) that is \( \mathcal{A} \)-creation-oblivious.

Then \( \epsilon_\delta(C_n, \tilde{C}_n^{\tilde{\beta}[i], \tilde{\beta}[i]}) = \prod_i \epsilon_\delta(C_n, \tilde{C}_n^{\tilde{\beta}[i], \tilde{\beta}[i]}) \) with \( \forall i \in [1, n], \tilde{\delta}^i = \text{oiblivious} \). Let \( \tilde{E}^\mathcal{A}[i], \tilde{\mathcal{A}}[i], \tilde{\epsilon}[i] \) \( \text{(complete, strong and bijective)} \) PCA executions-matching from \( \tilde{X} \).

\[ \text{Lemma 11.24 (measure after decomposition for oblivious creation scheduler).} \]

\[ \text{Theorem 11.25 (monotonicity).} \]

\[ \text{Proof.} \]

We recall the remark of definition 11.3 of \( \mathcal{A} \)-creation-oblivious scheduler for a \( \mathcal{A} \)-conservative PCA that raises the fact that if an execution fragment \( \tilde{\delta} \in \text{Frag}^* (\tilde{W}) \) verifying i) \( \tilde{\delta} \) ends on \( \mathcal{A} \)-creation, ii) \( W \cdot \mu_{\mathcal{A}+} (\tilde{\delta}) = e \) and iii) \( \text{trace}(\tilde{\delta}) = \beta \) exists, then \( \tilde{\delta}_\mathcal{A}, \tilde{\mathcal{A}}, \tilde{\epsilon} \) the sub-scheduler conditioned by \( \tilde{\delta} \) and \( \tilde{\delta} \) in the sense of definition 11.4. Then we simply apply lemma 11.5, which states that for every \( \alpha \in \text{A} \)-creation-oblivious and \( \alpha \in \text{Frag}^* (\tilde{E} || X) \), for \( \tilde{\delta}[\mathcal{A}] \), the sub-scheduler conditioned by \( \tilde{\delta} \) in schedulers(\( \tilde{E} || X \)) and \( \alpha \) (in the sense of definition 11.4), for \( \epsilon_\delta \) generated by \( \tilde{\delta}, \epsilon_\delta (\mathcal{A}_\alpha) = \epsilon_\delta (\mathcal{A}_\alpha) \cdot \epsilon_\delta (\mathcal{A}_\alpha) \) with \( \delta[\mathcal{A}], (\alpha_\alpha) = \delta(\alpha_\alpha, \alpha_\alpha) \) for every \( \alpha \) with \( f \text{state}(\alpha_\alpha) = \text{state}(\alpha_\alpha) \).

For every \( \alpha \in \prod C_n^{\tilde{\beta}[i], \tilde{\beta}[i]} \), for \( \tilde{a} = \alpha \)-decomposition, \( \epsilon_\delta (\mathcal{A}_\alpha) = \prod \epsilon_\delta (\alpha_\alpha) \cdot \epsilon_\delta (\alpha_\alpha) \) with \( \alpha_\alpha \) in \([1 : i - 1] = \alpha_\alpha \in \mathcal{A} \) and \( q_i = \text{start}(\tilde{E}^\mathcal{A}[i] \cdot X^i) \) where \( \tilde{E}^\mathcal{A}[i] \cdot X^i \) is defined as in lemma 11.22.

By \( \mathcal{A} \)-creation-oblivious property of \( \delta \), for every pair of vectors \( \alpha_\alpha \in [1 : i - 1] = \alpha_\alpha \in [1 : i - 1] \), s. t. \( \forall j \in [1 : i - 1] \), \( \text{trace}(\tilde{\delta}[\mathcal{A}] | j) \cdot (\tilde{E}^\mathcal{A}[i] \cdot X^i) = (\tilde{E}^\mathcal{A}[i] \cdot X^i) \cdot (\tilde{E}^\mathcal{A}[i] \cdot X^i) = e_j \). For every \( \pi \in \text{dom}(\tilde{E}^\mathcal{A}[i] \cdot X^i) \), \( \tilde{\delta}[\mathcal{A}] \cdot (\pi) = \tilde{\delta}[\mathcal{A}] \cdot (\pi) \). Hence, for every \( i \in [1, n] \) we note \( \tilde{\delta} \) in schedulers(\( \tilde{E}^\mathcal{A}[i] \cdot X^i \)) that matches \( \tilde{\delta}[\mathcal{A}] \cdot (\pi) \) on dom(\( \tilde{E}^\mathcal{A}[i] \cdot X^i, \tilde{\mathcal{A}}[i] \)).

Thus \( \epsilon_\delta (\mathcal{A}_\alpha) = \prod \epsilon_\delta (\alpha_\alpha) \cdot \epsilon_\delta (\alpha_\alpha) \) and by lemma 11.22.

\[ \epsilon_\delta (\tilde{C}_n^{\tilde{\beta}[i], \tilde{\beta}[i]}) = \prod_{i \in [1 : i - 1]} a_\alpha \in \mathcal{A}_\alpha \cdot \alpha_\alpha \in \prod C_n^{\tilde{\beta}[i], \tilde{\beta}[i]}) \]
In the remaining we note \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \mid ^{le} \) the automaton \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \mid ^{le} q \) as per definition 9.9 where \( q \) is the unique state of \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \) s. t. map(\( \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(q) \)) \( \mathcal{A} \) = \( q \mathcal{A} \) and \( \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(q) \setminus \{ \mathcal{A} \} = \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(\text{Isstate}(e)) \). Respectively, we note \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \mid ^{le} q \) the automaton \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \mid ^{le} q \) as per definition 9.9 where \( q \) is the unique state of \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \) s. t. map(\( \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(q) \)) \( \mathcal{A} \) = \( q \mathcal{A} \) and \( \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(q) \setminus \{ \mathcal{B} \} = \text{config}(\hat{\mathcal{E}} \mid X_{\mathcal{E}})(\text{Isstate}(e)) \). Finally, we note \( \hat{\mathcal{E}} \mid e \hat{\mathcal{E}} \mid ^{le} \text{Isstate}(e) \).

Let \( \hat{\sigma} \in \text{CroB}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \). We need to show that \( \hat{\sigma}' \in \text{CroB}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \) s. t. for every sequence of actions \( \beta \), for every \( e \in \text{Execs}(\hat{\mathcal{E}}) \), for \( \hat{\zeta} = (\beta, \hat{e}) \), \( \epsilon_{\beta} (C_{\mathcal{E}}^{\mathcal{E}}) = \epsilon_{\beta} (C_{\mathcal{E}}^{\mathcal{E}}) \) where \( C_{\mathcal{E}}^{\mathcal{E}} = \text{Class}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \), \( \text{print}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \), \( \hat{\mathcal{E}} \mid X_{\mathcal{E}} \).

\( \forall \beta \in \text{trace}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \cup \text{trace}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \), \( \forall e \in \text{Execs}(\hat{\mathcal{E}}) \), we note \( \sigma_{\beta,e} \) the \( (\hat{\mathcal{E}} \mid X_{\mathcal{E}})^{le} \), \( \mathcal{A}^{\beta} \) alter-ego of \( \hat{\sigma} \mid X_{\mathcal{E}} \).

Let \( e = \mu_{\mathcal{E}} \mathcal{C} (\mathcal{E}) \) We note \( \sigma_{\beta,e} = \hat{\sigma} \mid X_{\mathcal{E}} \) alter-ego of \( \sigma_{\beta,e} \).

We build \( \hat{\sigma}' \in \text{CroB}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \) as follows:

\( \forall \beta \in \text{trace}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \cup \text{trace}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \), \( \forall e \in \text{Execs}(\hat{\mathcal{E}}) \), \( \forall \beta', \pi' \in \text{Frag}^*(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \) s. t. \( \pi \) ends on \( \mathcal{B} \) creation, ii) \( (\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \mid \mathcal{B} = \hat{\mathcal{E}} \mid X_{\mathcal{E}} \), iii) \( \text{trace}(\hat{\mathcal{E}}) = \beta \), iv) \( \text{Isstate}(\hat{\mathcal{E}}) = \mathcal{F} \text{state}(\hat{\mathcal{E}}) \), \( \hat{\sigma}'(\hat{\pi} \hat{\pi}) \) = \( \hat{\sigma}'(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \).

\( \forall e \in \text{Execs}(\hat{\mathcal{E}}) \), \( \forall \beta, \beta' \in \text{Frag}(\hat{\mathcal{E}} \mid X_{\mathcal{E}}) \) balanced with \( \beta, \beta' \) s. t. \( \epsilon_{\beta}(\mathcal{E}) \).

We need to show that \( \epsilon_{\beta}(\mathcal{E}) = \epsilon_{\beta}(\mathcal{E}) \).

We apply lemma 11.23 to obtain:

\( \epsilon_{\beta}(\mathcal{E}) = \sum_{e \in \text{Execs}(\mathcal{E})} e \epsilon_{\beta}(\mathcal{E}) + \sum_{\mathcal{E} \hat{\epsilon} e} \sum_{\beta} \beta \epsilon_{\beta}(\mathcal{E}) \).

Since \( \mathcal{E} \) and \( \mathcal{E} \) are semantically equivalent, the sets \( \{ e \in \text{Execs}(\mathcal{E}) \mid e \} \) and \( \{ e \in \text{Execs}(\mathcal{E}) \mid e \} \) in bijection. Hence, it is enough to show that \( \forall (e, \mathcal{E}) \in \text{Execs}(\mathcal{E}) \times \text{Execs}(\mathcal{E}) \) with \( e = \mu_{\mathcal{E}} \mathcal{C} \) and \( \mathcal{E} \).

\( \epsilon_{\beta}(\mathcal{E}) = \epsilon_{\beta}(\mathcal{E}) \).

\( \epsilon_{\beta}(\mathcal{E}) = \epsilon_{\beta}(\mathcal{E}) \).

95
By definition, $\tilde{\sigma}$ is $\mathcal{A}$-creation-oblivious, and by construction, $\tilde{\sigma}$ is $\mathcal{B}$-creation-oblivious. This allows us to apply lemma 11.24 to obtain:

- $\epsilon_{A}(C_{\beta}[i],_\alpha\epsilon_{\tilde{\sigma}}[i]) = \prod_{i=1}^{n} \epsilon_{\tilde{\sigma}}(C_{\beta}[i],_\alpha\epsilon_{\tilde{\sigma}}[i])$ with $\forall i \in [1, n], \tilde{\sigma}[i] = oblivious\rightarrow\mathcal{A}, \beta[i], _\epsilon[\alpha][i] (\tilde{\sigma})$.

- $\epsilon_{B}(C_{\beta}[i],_\beta\epsilon_{\tilde{\sigma}}[i]) = \prod_{i=1}^{n} \epsilon_{\tilde{\sigma}}(C_{\beta}[i],_\beta\epsilon_{\tilde{\sigma}}[i])$ with $\forall i \in [1, n], \tilde{\sigma}[i] = oblivious\rightarrow\mathcal{B}, \beta[i], _\epsilon[\beta][i] (\tilde{\sigma})$.

- where $\tilde{\epsilon}[i] = \tilde{\epsilon}[1] \sim \cdots \sim \tilde{\epsilon}[i-1]$ for $\tilde{\epsilon} \in \{\beta, _\epsilon[\beta], _\epsilon[\alpha]\}$

- $\tilde{C}_{\mathcal{A}}(\tilde{\beta}[i], _\alpha\epsilon_{\tilde{\sigma}}[i]) = class(Exec(\tilde{C}[i]|X_{A}),_\alpha\epsilon_{\tilde{\sigma}}[i], print_{\mathcal{A}}(\tilde{C}[i],_\beta\epsilon_{\tilde{\sigma}}[i]))$

- $\tilde{C}_{\mathcal{B}}(\tilde{\beta}[i], _\beta\epsilon_{\tilde{\sigma}}[i]) = class(Exec(\tilde{C}[i]|X_{B}),_\beta\epsilon_{\tilde{\sigma}}[i], print_{\mathcal{B}}(\tilde{C}[i],_\epsilon[\beta][i]))$

Thus it is enough to show that $\forall i \in [1, n], \epsilon_{\tilde{\sigma}}(C_{\tilde{\sigma}[i],_\alpha\epsilon_{\tilde{\sigma}}[i]} = \epsilon_{\tilde{\sigma}^{*}}(C_{\tilde{\sigma}[i],_\beta\epsilon_{\tilde{\sigma}}[i]}).$ Let $i \in [1, n]$

We can apply lemma 11.12 to obtain:

- $\epsilon_{\tilde{\sigma}}(C_{\tilde{\beta},_\alpha\epsilon_{\tilde{\sigma}}[i]}) = \epsilon_{\tilde{\sigma}}(C_{\tilde{\beta}[i],_\alpha\epsilon_{\tilde{\sigma}}[i]})$

- $\epsilon_{\tilde{\sigma}}(C_{\tilde{\beta},_\beta\epsilon_{\tilde{\sigma}}[i]}) = \epsilon_{\tilde{\sigma}}(C_{\tilde{\beta}[i],_\beta\epsilon_{\tilde{\sigma}}[i]})$

where:

- $\tilde{C}_{\mathcal{A}}(\tilde{\beta}[i], _\alpha\epsilon_{\tilde{\sigma}}[i]) = class(Exec(\tilde{C}[i]|X_{A}),_\alpha\epsilon_{\tilde{\sigma}}[i], print_{\mathcal{A}}(\tilde{C}[i],_\beta\epsilon_{\tilde{\sigma}}[i]))$ and

- $\tilde{C}_{\mathcal{B}}(\tilde{\beta}[i], _\beta\epsilon_{\tilde{\sigma}}[i]) = class(Exec(\tilde{C}[i]|X_{B}),_\beta\epsilon_{\tilde{\sigma}}[i], print_{\mathcal{B}}(\tilde{C}[i],_\epsilon[\beta][i]))$

- $\sigma_{\alpha, \beta}[i]$ is the $((\tilde{C}[i]|X_{A}),_\alpha\epsilon_{\tilde{\sigma}}[i], \tilde{\sigma})$-alter-ego of $\tilde{\sigma}$.

- $\sigma_{\beta}[i]$ is the $((\tilde{C}[i]|X_{B}),_\beta\epsilon_{\tilde{\sigma}}[i], \tilde{\sigma})$-alter-ego of $\tilde{\sigma}$.

Hence it is sufficient to show that $\epsilon_{A}(C_{\tilde{\beta},_\alpha\epsilon_{\tilde{\sigma}}[i]}) = \epsilon_{\tilde{\sigma}}(C_{\tilde{\beta}[i],_\alpha\epsilon_{\tilde{\sigma}}[i]}) = \epsilon_{\tilde{\sigma}^{*}}(C_{\tilde{\beta}[i],_\beta\epsilon_{\tilde{\sigma}}[i]})$.

Finally, we find again our construction (*):
This lead us to 
\[ \sigma \rightarrow_{[\mathcal{A} \beta \gamma \delta]} \sigma \] 
\[ \epsilon_{\mathcal{C}} \rightarrow_{[\mathcal{A} \beta \gamma \delta]} \epsilon_{\mathcal{C}} \] 
\[ \mathcal{C} \rightarrow_{[\mathcal{A} \beta \gamma \delta]} \mathcal{C} \] 
which ends the proof. 

\[ \square \]

12 TASK SCHEDULE

We have shown in previous section that \( S_{\text{prob}}^{\text{prob}} \) was a monotonic relationship. In this section, we explain why an easy to use off-line scheduler introduced by Canetti & al. [4] is not creation-oblivious which surprisingly prevent us to obtain monotonicity of implementation for scheduler schema.

12.1 Task-schedule

Here we present a subclass of fully off-line schedulers, called task-schedules. Before explaining what is a task-schedule, we introduce the definition of task, adapted from [2] to the dynamic setting. In practice a task is invoked by the scheduler to potentially trigger an action from a sub-component of the automaton. An automaton can be the result of the composition of several automata, themselves the result of the composition of several automata and so on and forth. We assume the existence of a subset \( \text{Autids}_0 \subset \text{Autids} \) that represents the "atomic entities" of our formalism. Any automaton is the result of the composition of automata in \( \text{Autids}_0 \).

Definition 12.1 (Constitution). For every \( \mathcal{A} \in \text{Autids} \), we note 
\[ \text{constitution}(\mathcal{A}) : \begin{cases} \text{states}(\mathcal{A}) & \rightarrow \mathcal{P}(\text{Autids}_0) \text{ where } \mathcal{P}(\text{Autids}_0) \text{ denotes the power set of } \text{Autids}_0 \\ q & \rightarrow \text{constitution}(\mathcal{A})(q) \end{cases} \]
For every \( \mathcal{A} \in \text{Autids}_0 \), for every \( q \in \text{states}(\mathcal{A}) \), \( \text{constitution}(\mathcal{A})(q) = \{ \mathcal{A} \} \).
For every \( \mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n) \in (\text{Autids}_0)^n \), for every \( q \in \text{states}(\mathcal{A}) \) with \( \mathcal{A} = \mathcal{A}_1 || \ldots || \mathcal{A}_n \), \( \text{constitution}(\mathcal{A})(q) = \mathcal{A} \).

The constitution of a PCA is defined recursively through its configuration. For every PCA \( X \), for every \( q \in \text{states}(X) \), if we note \( (A,S) = \text{config}(X)(q) \), \( \text{constitution}(X)(q) = \bigcup_{\mathcal{A} \in A} \text{constitution}(\mathcal{A})(S(\mathcal{A})) \).

Definition 12.2 (Task). A task \( T \) is a pair \( (id, actions) \) where \( id \in \text{Autids}_0 \) and \( actions \subset \text{actions}(\text{aut}(id)) \) is a set of action labels. Let \( T = (id, actions) \), we note \( id(T) = id \) and \( actions(T) = actions \).

Definition 12.3 (Enabled task). Let \( \mathcal{A} \in \text{Autids} \). A task \( T \) is said enabled in state \( q \in \text{states}(\mathcal{A}) \) if:

- \( id(T) \in \text{constitution}(\mathcal{A})(q) \)
- It exists a unique local action \( a \in \text{loc}(\mathcal{A})(q) \cap actions(T) \) (noted \( a \in T \) to simplify) enabled at state \( q \) (that is it exists \( \eta \in \text{Disc}(Q,\mathcal{A}) \) s. t. \( (q,a,\eta) \in D,\mathcal{A} \).

In this case we say that \( a \) is triggered by \( T \) at state \( q \).

Now we are ready to define a task-schedule, which is a particular subclass of schedulers.

We are not dealing with a task-schedule of a specific automaton anymore, which differs from [2]. However the restriction of our definition to "static" setting matches their definition.

Definition 12.4 (task-schedule). A task-schedule \( \rho = T_1, T_2, T_3, \ldots \) is a (finite or infinite) sequence of tasks.

Since our task-schedule is defined, we are ready to solve the non-determinism and define a probability on the executions of a PSIOA. We use the measure of [2].

Definition 12.5. (task-based probability on executions: \( \text{apply}_\mathcal{A}(\mu, \rho) : \text{Frags}(\mathcal{A}) \rightarrow [0,1] \)) Let \( \mathcal{A} \) be a PSIOA. Given \( \mu \in \text{Disc}(\text{Frags}(\mathcal{A})) \) a discrete probability measure on the execution fragments and a task schedule \( \rho, \text{apply}(\mu, \rho) \) is a probability measure on \( \text{Frags}(\mathcal{A}) \). It is defined recursively as follows.
(1) apply\(_A\)(\(\mu, \lambda\)) := \mu. Here \(\lambda\) denotes the empty sequence.

(2) For every \(T\) and \(\alpha \in \text{Frags}^*(\mathcal{A})\), apply\(_A(T)(\alpha) := p_1(\alpha) + p_2(\alpha)\), where:

\[
p_1(\alpha) = \begin{cases} 
\mu(\alpha')\eta_1(\alpha, q', a)(q) & \text{if } \alpha = \alpha' - (a, q), q' = \text{Istate}(\alpha') \text{ and } a \text{ is triggered by } T \text{ enabled after } \alpha' \\
0 & \text{otherwise}
\end{cases}
\]

\[
p_2(\alpha) = \begin{cases} 
\mu(\alpha) & \text{if } T \text{ is not enabled after } \alpha \\
0 & \text{otherwise}
\end{cases}
\]

(3) If \(\rho\) is finite and of the form \(\rho'T\), then apply\(_A(\mu, \rho) := \text{apply}_A(\mu, \rho')\), \(T\).

(4) If \(\rho\) is infinite, let \(\rho_i\) denote the length-\(i\) prefix of \(\rho\) and let \(\text{pm}_i\) be apply\(_A(\mu, \rho_i)\). Then apply\(_A(\mu, \rho) := \lim_{i \to \infty} \text{pm}_i\).

**Proposition 2.** Let \(\mathcal{A}\) be a PSIOA. For each measure \(\mu\) on Frags\(^*(\mathcal{A})\) and task schedule \(\rho\), there is a scheduler \(\sigma\) for \(\mathcal{A}\) such that apply\(_A(\mu, \rho)\) is the generalized probabilistic execution fragment \(\epsilon_{\sigma, \mu}\).

**Proof.** The result has been proven in [2], appendix B.4.

---

Fig. 31. Non-deterministic execution: The scheduler allows us to solve the non-determinism, by triggering an action among the enabled one. We give an example with an automaton \(\mathcal{A} = (Q, \delta, q_0, \sigma, \text{Istate}(\mathcal{A}), D)\) and the tasks \(T_0, T_1, T_2\) (for green, orange, pink, blue) with the respective actions \(\{a\}, \{d\}, \{b, b'\}, \{c, c'\}\), and the tasks \(T_0, T_2\) with the respective actions \(\{a, b\}, \{c, c', d\}\). At state \(q_0\), \(\sigma(\mathcal{A})(q_0) = (\emptyset, \{a\}, \{d\})\). Hence both \(a\) and \(d\) are enabled local action at \(q_0\), which means both \(T_0\) and \(T_2\) are enabled at state \(q_0\), but \(T_0\) is not enabled at state \(q_0\) since it does not solve the non-determinism \((a\) and \(d\) are enabled local action at \(q_0\)). At state \(q_1\), \(T_0\) is enabled but neither \(T_0\) nor \(T_2\). We give some results: apply\(_A(\delta_{q_0}, T_0)(q^0, a, q^1) = 1\)

apply\(_A(\delta_{q_0}, T_0)(q^0, a, q^1, b, q^2) = \frac{1}{2}\)

apply\(_A(\delta_{q_0}, T_0)(q^0, a, q^1, b, q^2, c, q^3) = \frac{3}{8}\)

apply\(_A(\delta_{q_0}, T_0, T_2)(q^0, a, q^1, b, q^2, c, q^3) = 3/8\), since \(T_0\) is not enabled at state \(q^3).\)
12.2 Why a task-scheduler is not creation-oblivious?

Let imagine the following example. The class $C^x$ is composed of two executions $\alpha^{x,1}$ and $\alpha^{x,2}$, the class $C^y$ is composed of two executions $\alpha^{y,1}$ and $\alpha^{y,2}$ and the class $C^z$ is composed of four executions $\alpha^{z,11} = \alpha^{x,1} \bowtie \alpha^{y,1}$, $\alpha^{z,12} = \alpha^{x,1} \bowtie \alpha^{y,2}$, $\alpha^{z,21} = \alpha^{x,2} \bowtie \alpha^{y,1}$, $\alpha^{z,22} = \alpha^{x,2} \bowtie \alpha^{y,2}$. Let $\rho = \rho^1 \bowtie \rho^2$ be a task-schedule. We do not have $\text{apply}(\cdot, \rho)(C^z) = \text{apply}(\cdot, \rho^1)(C^x) \cdot \text{apply}(\cdot, \rho^2)(C^y)$! Indeed, the executions $\alpha^{x,1}$ and $\alpha^{x,2}$ can differ s.t. they do not ignore the same tasks. Typically, $\rho^1$ could be written $\rho^1 = \rho^{1,a} \bowtie \rho^{1,b}$ where the last action of $\alpha^{x,1}$ is triggered by the last task of $\rho^{1,a}$ and $\rho^{1,b}$ is ‘ignored by $\alpha^{x,1}$. The issue comes if both $\text{apply}(\cdot, \rho^1)(C^y) \neq \emptyset$ and $\text{apply}(\cdot, \rho^{1,b} \bowtie \rho^{1,b})(C^y) \neq \emptyset$. The point is that $C^z$ can be obtained with different cut-paste: cut-paste $A$: $\rho^{1,a}$ for $C^x$ and $\rho^{1,b} \bowtie \rho^2$ for $C^y$; cut-paste $B$: $\rho^1$ for $C^x$ and $\rho^2$ for $C^y$.

13 CONCLUSION

We extended dynamic I/O Automata formalism of Attie & Lynch [1] to probabilistic settings in order to cope with emergent distributed systems such as peer-to-peer networks, robot networks, adhoc networks or blockchains. Our formalism includes operators for parallel composition, action hiding, action renaming, automaton creation and use a refined definition of probabilistic configuration automata in order to cope with dynamic actions. The key result of our framework is as follows: the implementation of probabilistic configuration automata is monotonic to automata creation and destruction. That is, if systems $X_A$ and $X_B$ differ only in that $X_A$ dynamically creates and destroys automaton $A$ instead of creating and destroying automaton $B$ as $X_B$ does, and if $A$ implements $B$ (in the sense they cannot be distinguished by any external observer), then $X_A$ implements $X_B$. This result is particularly interesting in the design and refinement of components and subsystems in isolation. In our construction we exhibit the need of considering only creation-oblivious schedulers in the implementation relation, i. e. a scheduler that, upon the (dynamic) creation of a sub-automaton $A$, does not take into account the previous internal actions of $A$ to output (randomly) a transition.

Interestingly and of independent interest, motivated by the monotonicity of execution w.r.t. to automata creation, we introduce new proof techniques to deduce certain properties of a system $X_A$ from a sub-automaton $X_A$ dynamically created and destroyed by $X_A$. This proof technique is used to construct a homomorphism between the probabilistic spaces of automata executions. Then we expose such homomorphism from a system $X_A$ to a new system resulting from the composition of $A$ and $X_A \setminus \{A\}$. The latter corresponds intuitively to the system $X_A$ deprived of $A$. Furthermore, the homomorphism is used to show that under certain minor technical assumptions, if $X_A$ and $X_B$ differ only in the fact that $X_A$ dynamically creates and destroys the automaton $A$ instead of creating and destroying the automaton $B$ as $X_B$ does, then $X_A \setminus \{A\}$ and $X_B \setminus \{B\}$ are semantically equivalent, i.e. they only differ syntactically. The homomorphism is finally reused to establish the monotonicity of the implementation relation. Our technique can be used in extensions of our formalism with time and cryptography notions.

As future work we plan to extend the composable secure-emulation of Canetti et al. [4] to dynamic settings. This extension is necessary for formal verification of protocols combining probabilistic distributed systems and cryptography in dynamic settings (e.g. blockchains, secure distributed computation, cybersecure distributed protocols etc).

REFERENCES


