Université de Sherbrooke
Département d'informatique

Technical report

Verification of Parameterized Algebraic State-Transition Diagrams

October 3, 2017

Raphaël Chane-Yack-Fa

Abstract

In this article, we study the information system verification problem as a parameterized verification one. Informations systems are modeled as multiparameterized systems in a formal language based on the ASTD notation. Then, we use the WSTS theory to solve the coverability problem for an unbounded ASTD state space. Moreover, we define a new framework to prove the effective pred-basis condition of WSTSs.

Contents

Chapter 1

Verification of Parameterized Algebraic State-Transition Diagrams

1.1 Introduction

Information Systems (IS) have been evolving for years now and the validation of such systems has become a prominent topic. An IS can be viewed as a complex system composed of a set of related entities and which can manage a substantial amount of data. They are used for various applications such as banking services, online sales or data warehouse [5]. In some cases, they are critical systems and one must ensure that the system is reliable. Therefore, the verification of IS specifications is essential. Furthermore, in this paper, we will see IS as parameterized systems which model systems with an unbounded number of components. For instance, the specification of a library management system, which is an IS, may abstract the number of books which interact in the library, and thus can be represented by a parameterized system. The aim of this paper is to find suitable techniques to specify and verify ISs from the point of view of parameterized systems.

Parameterized verification is a widely addressed topic in the literature [32, 16, 10, 33, 40, 39, 25, 29, 2, 11, 30] and we can explore the relation between parameterized systems and ISs. Among the characteristics of ISs, we can notice the multiplicity of entity types and the complex relationships between them. For example, in the library system we can model a book entity and a member entity together with a loan relationship. The number of instances of each entity is unbounded: that justifies the need for a parameterized specification. In general, an IS has more than one entity and thus multiple parameters. However, in most of the methods in the literature, parameterized verification is limited to some specific models like systems with only one parameter or without any relationship. We propose in this paper a method that handles systems with many parameters and relationships. To model ISs, we use an extension of the *Algebraic State-Transition Diagram* (ASTD) notation, proposed in [24], which we call *Parameterized ASTD* (PASTD) and which can model many related entities. As for the verification we based our method on the *Well Structured Transition Systems* (WSTS) framework [18] which can verify monotone infinite systems equipped with a well-quasi-order.

In order to use the WSTS framework, we make some hypothesis on ISs that are essential to prove the monotony of the system and the termination of the verification procedure. First, we assume that the specification of an IS cannot synchronize an arbitrary number of processes as it would break the monotony condition, which states that a similar configuration with more entity

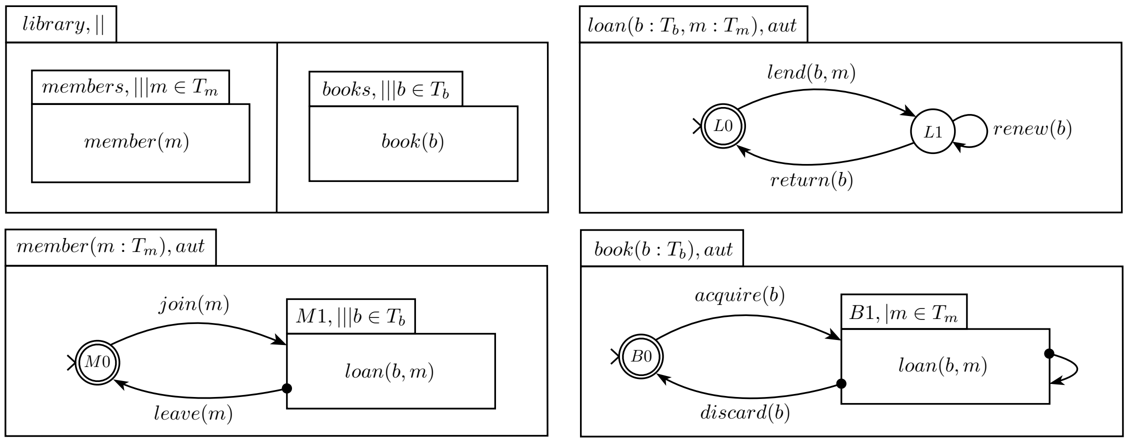


Figure 1.1: Example of a library system

instances must be able to fire a similar transition. Moreover, to ensure termination, we devise some structural conditions on the states. However, unlike [18, 11, 33, 30], we deal with those conditions separately as we describe a general semi-decision procedure working for a large number of cases. This leads us to the definition of a new framework called *Ranked Monotone Transition System* (RMTS) which is useful to show the effectiveness condition of WSTS. As a result, we prove that PASTDs are RMTSs.

This paper is structured as follows. Section 1.2 presents the ASTD notation and its PASTD extension. In Section 1.3, we recall the WSTS theory and show how to apply it to PASTDs by determining adequate conditions. Section 1.4 presents the RMTS framework and proves that PASTDs are RMTSs. The related work is presented in Section 1.5 and Section 1.6 concludes.

1.2 A visual process algebra

An *Algebraic State-Transition Diagram* (ASTD) [24] is a graphical notation combining automata, statecharts and process algebras to describe complex dynamic systems like information systems. ASTDs are closely related to process algebras like CSP [27], CCS [34], ACP [4], LOTOS [7] and EB^3 [21]. Essentially, they are like a process algebra with hierarchical automata as elementary process expressions. Automata can be combined freely with process algebra operators. ASTDs have a structured operational semantics in the Plotkin style, which was first used by Milner for CCS and later on for LOTOS and CSP [37]. They are recursively defined structures and include many types like automaton, synchronization, quantified choice and quantified interleaving.

ASTDs are useful to model information systems as they provide a concise, visual and formal mechanism for specifying all the scenarios of an information system [24]. For example, they make explicit the handling of instances of information system entities by using quantifications. Furthermore, the model has been used in [15] to specify access control and security policies.

For instance, an ASTD model of a library system is given in Figure 1.1. The system manages loans of books by members. Books and members are the entities of the library system. The ASTD on the top left hand corner, whose name is *library*, is a synchronization between two other ASTDs, which consist in a quantified interleaving on the set T_m of members for the the process *member*, and a quantified interleaving on the set T_b of books for the process *book*. The process *book*(b) is described by the ASTD on the bottom right corner. A book is acquired by the library. It can be discarded if it is not lent. If acquired, a book b can "choose" a member m to run the process $loan(b, m)$. The member process is similar except that a member m runs the $loan(b, m)$ process for every book.

1.2.1 Parameterized ASTD

In this paper, we will restrict ourselves to a fragment of the ASTD language. The graphical and textual notation of the original ASTDs are detailed in [23].

Let us denote a labeled tree by an expression thanks to the grammar $Tree ::= Node \mid Node[Tree, ..., Tree]$.

Definition 1 (ASTD expression)**.** *ASTD expressions are defined inductively by the following grammar:*

$$
\begin{array}{lll}
\boldsymbol{F} ::= \mathcal{A} & & \textit{(automaton)} \\
\quad \mid \boldsymbol{F} \,||_{\Delta}\, \boldsymbol{F} & & \textit{(synchronization)} \\
\quad \mid \,|_{x \in T} \boldsymbol{F} & & \textit{(quantified choice)} \\
\quad \mid \,|||_{x \in T} \boldsymbol{F} & & \textit{(quantified interleaving)}
\end{array}
$$

where:

- *$\mathcal{A}$ is an Automaton ASTD $(Q, \Sigma, \delta, q_0, Q_F)$ such that Q is a finite set of states where for each $q \in Q$, q is either an elementary state or a composite state,* i.e. *another ASTD expression, Σ a set of labels, δ a labeled transition relation, $q_0 \in Q$ an initial state, $Q_F \subseteq Q$ a set of final states; note that a transition can be final (represented by a big dot at the origin of the arrow),* i.e. *it can be fired only from final states of the component ASTDs, or non-final,* i.e. *it can be fired at any time;*

- *$\boldsymbol{F} \,||_{\Delta}\, \boldsymbol{F}$ denotes a Synchronization between two component ASTDs running concurrently by executing events, whose labels are in Δ, at the same time and interleaving the other events; if Δ is empty we denote the operator by $|||$ and if Δ is omitted, the processes synchronize on the set of shared labels;*

- *a Quantified Choice ASTD $|_{x \in T} \boldsymbol{F}$ allows us to pick a value v from a finite set T and execute its component ASTD $\boldsymbol{F}$, where every occurrence of the variable x is replaced by the value v;*

- *a Quantified Interleaving ASTD $|||_{x \in T} \boldsymbol{F}$ allows us to execute as many interleaving instances of $\boldsymbol{F}$ as the number of values in T, where each instance is executed such that x is replaced by the corresponding value.*

In the following, we will also use the following tree-like notation, which will be easier to manipulate:

$$
\begin{array}{lll}
\boldsymbol{F} ::= & \mathcal{A}[q_1[\boldsymbol{F'}], ..., q_k[\boldsymbol{F'}]] & \quad \textit{(one subtree for each state } q_i \in Q\textit{)} \\
\mid & ||_{\Delta}[\boldsymbol{F}, \boldsymbol{F}] & \\
\mid & |_{x \in T}[\boldsymbol{F}] & \\
\mid & |||_{x \in T}[\boldsymbol{F}] &
\end{array}
$$

$$
\begin{array}{ll}
\boldsymbol{F'} ::= & \boldsymbol{F} \\
\mid & \epsilon
\end{array}
$$

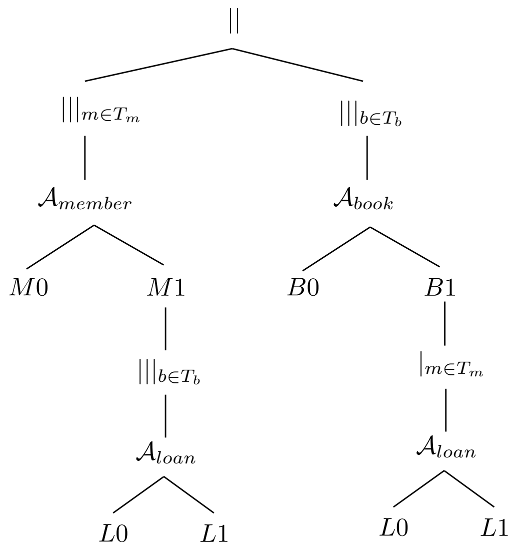


Figure 1.2: The tree representation of an ASTD

Note that ϵ is used to create elementary automaton states. Figure 1.2 shows the tree notation of the ASTD of Figure 1.1, where $\mathcal{A}_{member}$, $\mathcal{A}_{book}$ and $\mathcal{A}_{loan}$ are the automata for member, book and loan processes respectively.

In this fragment of the ASTD language, we make two notable modifications from the original definition. First, we do not allow recursion, so that an ASTD specification is a tree-like structure. Moreover, we do not allow quantified synchronizations and we weaken the Quantified Interleaving ASTD by allowing the process to be in a final state at any time, *i.e.* there is no final synchronization between all interleaving processes. Indeed, the property of monotony, that we will explain in Section 1.3.2, is easier to obtain without quantified synchronizations.

The quantified operators (interleaving and choice) allow us to model replication of processes and complex interactions between them. For instance, in the example of the library system, we use the quantified interleaving to model many replicated processes running concurrently. To model a library that works with any number of members and books, we can consider a more abstract specification that takes the sets of members and books as parameters. Indeed, we allow quantification sets to be variables in quantified choice et quantified interleaving. We call those variables the parameters of the system.

Definition 2 (Parameterized ASTD)**.** *A Parameterized ASTD (PASTD) is a triple* $(F, \overrightarrow{T}, \overrightarrow{P})$ *where* F *is an ASTD expression,* $\overrightarrow{T} = (T_1, ..., T_n)$ *is a vector of* n *variables, called parameters, and* $\overrightarrow{P} = (P_1, ..., P_n)$ *is a vector of* n *sets of elements, called parameter domains. Each variable* T_i *represents a finite subset of the possibly infinite set* P_i *and appears in the expression of* F *as quantification set for quantified choice or quantified interleaving.*

In a PASTD $(F, \overrightarrow{T}, \overrightarrow{P})$, remark that the ASTD F does not correspond to a concrete transition system as it contains the variables $\overrightarrow{T}$. However, if we choose a sequence of sets $R_1 \subseteq P_1, ..., R_n \subseteq P_n$, the substitution of $\overrightarrow{T}$ by $\overrightarrow{R}$ in F does represent a transition system. Intuitively, a PASTD represents the union of all possible instantiations of the expression F, which may correspond to an infinite system. For instance, in the library system, we use two parameters T_m and T_b, with parameter domains $P_m = \{m_1, m_2, ...\}$ and $P_b = \{b_1, b_2, ...\}$, respectively the sets of member and book identifiers.

As a PASTD describes a dynamical system, for each PASTD, we can define a set of PASTD states. A state will be given by a tree structure.

Definition 3 (ASTD state)**.** *ASTD states are defined inductively by the following grammar:*

$\boldsymbol{S}$ ::=	$(\mathsf{aut}_\circ, q)$	*(automaton, elementary state q)*
$\mid$	$(\mathsf{aut}_\circ, q)[\boldsymbol{S}]$	*(automaton, composite state q)*
$\mid$	$\vert\vert_\circ[\boldsymbol{S}, \boldsymbol{S}]$	*(synchronization)*
$\mid$	$\vert\!:_\circ$	*(quantified choice, value not chosen yet)*
$\mid$	$\vert\!:_\circ[p[\boldsymbol{S}]]$	*(quantified choice, p is chosen)*
$\mid$	$\vert\vert\vert\!:_\circ[p_1[\boldsymbol{S}], ..., p_k[\boldsymbol{S}]]$	*(quantified interleaving on* $\{p_1, ..., p_k\}$, $k \in \mathbb{N}_1$*)*

For a PASTD A, we denote by $\mathcal{T}_A$ the set of all well-formed states of A.

An ASTD state is represented by a tree where each node is labeled either by the type of state or by a value from quantification sets. If a type of state includes a sub-state in its definition, then it is represented by a child node. Furthermore, we split up the nodes for quantifications so that the values appear in child nodes.

Let us denote a graph by a pair (V, E) such that V is a set of vertices and E a set of edges, where an edge is a pair of vertices. We call labeled graph a triple (V, E, λ) where (V, E) is a graph and $\lambda : V \to \Lambda$ a labeling function with Λ a set of labels. A labeled tree can be given by an expression $Tree ::= Node \mid Node[Tree, ..., Tree]$ or equivalently by a labeled acyclic connected graph (V, E, λ). In the following, we will sometimes represent an ASTD state by a graph (V, E, λ). We denote by $Im(f)$ the image of the function $f : X \to Y$, *i.e.* $Im(f) = \{y \in Y \mid \exists x \in X \cdot f(x) = y\}$.

Definition 4. *A PASTD state is always associated with an ASTD expression with which it must be consistent. For a PASTD $A = (F, (T_1, ..., T_n), (P_1, ..., P_n))$ and a state $s = (V, E, \lambda) \in \mathcal{T}_A$, we denote by $val(s) = (R_1, ..., R_n)$ the sets of elements appearing in the state s,* i.e. *$R_i = P_i \cap Im(\lambda)$ for all $i \in 1..n$.*

For instance, consider the library system $(F, (T_m, T_b), (P_m, P_b))$ of Figure 1.1. Let the state s consist of two members m_1 and m_2 and two books b_1 and b_2, where the book b_2 is borrowed by the member m_1 as depicted in Figure 1.3. We have $val(s) = (\{m_1, m_2\}, \{b_1, b_2\})$. In this state, the member m_1 has joined the library and has borrowed the book b_2. The member m_2 has not joined the library yet; similarly, the book b_1 has not been acquired yet; these two cases are represented by the initial state of the member and book automata, respectively.

Remark that we allow an ASTD state to represent a more abstract state by omitting some branches of the quantified interleaving and by considering that the local configuration of some entities is unknown. See for example Figure 1.4 where the book b_1 is omitted in the quantified interleave of the state $M1$ in member process m_1.

The operational semantics of PASTD consists of a set of inference rules defined inductively on the ASTD states. For a PASTD A, we denote the corresponding transition relation on $\mathcal{T}_A$ by $\to_A$ and we obtain the transition system $\mathcal{S}_A = (\mathcal{T}_A, \to_A)$. Figure 1.5 shows an example of transition in the library system where the member m_1 returns the book b_2. For instance, $\vert\vert_\circ$ makes a transition if both its sub-branches are able to do the transition on the same event. $\vert\vert\vert_{:\circ}$ can do the transition if one of its branches can and $\vert_{:\circ}$ if its sub-branch can. Finally, an automaton state fires a transition

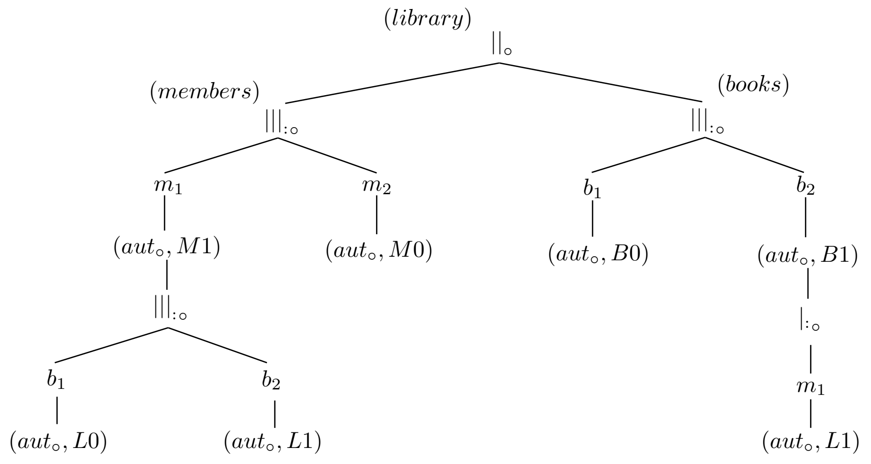


Figure 1.3: A state of the library system

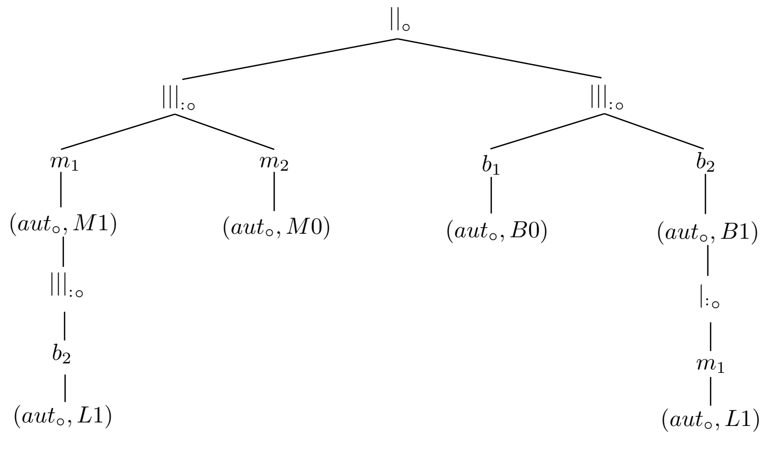


Figure 1.4: An abstract state

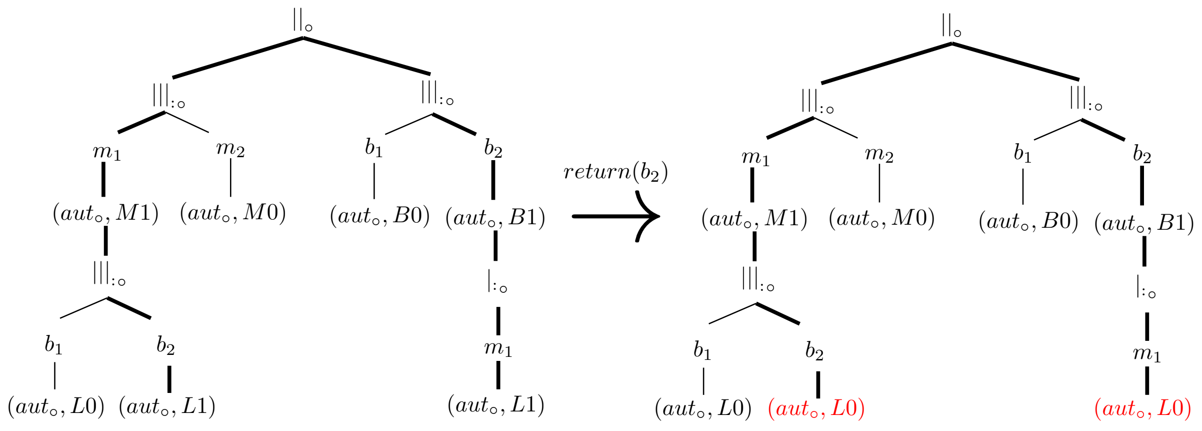


Figure 1.5: A transition

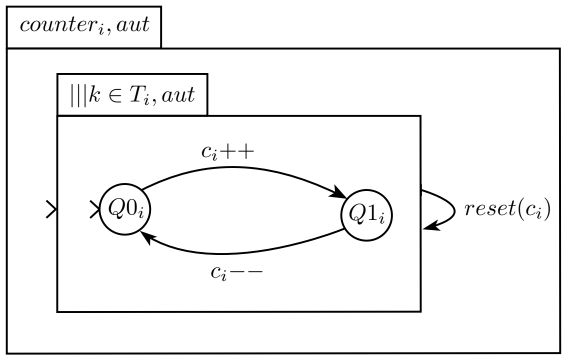


Figure 1.6: ASTD model of a counter

either by a classical automaton transition or within a sub-state. The branches involved in the transition of Figure 1.5 are depicted by bold strokes and the notable change in red. For a complete description of the operational semantics of ASTDs and PASTDs, see [23] and Appendix A.

1.2.2 Expressiveness of PASTD

Considering the previous extension of ASTDs with parameters, we know that PASTDs are more expressive than some types of infinite systems like *Petri Nets* [35] or *Vector Addition Systems with States* (VASS) [28]. More precisely, we remark that PASTDs can simulate VASSs with resets.

An n-dimensional VASS with resets (RVASS) is a finite directed graph (V, E) with arcs labeled by vectors of integers or a reset symbol r, *i.e.* $E \subseteq V \times (\mathbb{Z} \cup \{r\})^n \times V$ together with an initial vertex $p_i \in V$ and an initial natural vector $u_i \in \mathbb{N}^n$. A configuration is a pair $(p, u) \in V \times \mathbb{N}^n$. There is a transition between two configurations $(p, (u_1, ..., u_n)) \to (p', (u'_1, ..., u'_n))$ if $(p, (a_1, ..., a_n), p') \in E$, where for all $i \in 1..n$, either $a_i = u'_i - u_i$ or $a_i = r \wedge u'_i = 0$.

We denote a transition system by a pair $S = (Q, \to)$, where Q is a set of states and $\to \subseteq Q \times Q$ is the set of transitions between states. We write $q \to q'$ for $(q, q') \in \to$, and $q \xrightarrow{*} q'$ if either $q = q'$ or there exists a finite sequence of transitions $q \to q_1 \to q_2 \to ... \to q'$, called a path. We say that a transition system $(Q, \to)$ simulates another transition system $(Q', \hookrightarrow)$ if there exists an injection

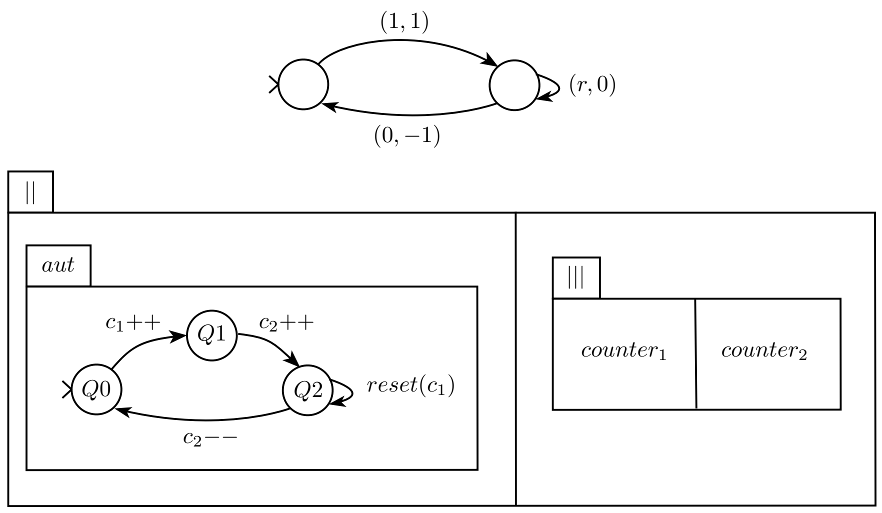


Figure 1.7: A RVASS and its PASTD simulation

$f : Q' \to Q$ such that for all $q'_1, q'_2 \in Q'$ such that $q'_1 \hookrightarrow q'_2$, we have $f(q'_1) \xrightarrow{*} f(q'_2)$.

Intuitively, an RVASS consists of a finite set of counters $\{c_1, ..., c_n\}$ and of a control state graph. Each arc of the graph is labeled by an action that can change the values of the counters c_i.

Proposition 1. *PASTDs simulate VASSs with resets.*

Proof. Let (V, E) be an RVASS. Let us construct a PASTD that simulates (V, E).

- First, let us model the counters. Each counter c_i can be modeled by a two states automaton ASTD within a quantified interleaving ASTD as shown in Figure 1.6. There is a $c_i{+}{+}$ action which increments the counter i, a $c_i{-}{-}$ action which decrements it and a reset action $reset(c_i)$ which sets c_i to zero. To handle the reset action, we add a non-final loop on the interleaving operation. The value of c_i is given by the number of processes in the local state $S1_i$. For an unbounded counter, the quantification set is simply represented by a variable T_i taking values in the subsets of $\mathbb{N}$.

- Second, the control graph can be represented by a simple automaton ASTD. According to the previous model of counters, we will increase or decrease only one counter at the same time and only by one. Thus, we need to split some transitions into some more complex sequences of transitions by adding new control states.

- Finally, The global PASTD modeling the RVASS is a synchronization between the control graph PASTD and every counter PASTD.

That way, each transition of configuration of the RVASS is mimicked by a transition or a sequence of transitions in the resulting PASTD. Thus, PASTDs can simulate RVASSs. See an example of simulation in Figure 1.7. □

This construction allows us to deduce some decidability results with regards to reachable configurations. For instance, the reachability problem is known to be undecidable for RVASSs [14]. Since simulation preserves reachability of states, we can conclude that the reachability problem is also undecidable for PASTDs.

Proposition 2. *Reachability is undecidable for PASTDs.*

Proof. PASTDs simulates RVASSs and reachability is undecidable for RVASSs. □

1.3 Well Structured PASTD

1.3.1 Preliminaries

For a transition system $S = (Q, \rightarrow)$, we define $Pred(q) = \{q' \in Q \mid q' \rightarrow q\}$ as the set of immediate predecessors of q and $Pred^*(q) = \{q' \in Q \mid q' \xrightarrow{*} q\}$ as the set of all predecessors of q.

A *quasi-ordering (qo)* is a reflexive and transitive binary relation $\leq$ on a set X; we also say that $(X, \leq)$ is a qo. A *partial ordering (po)* is an antisymmetric qo. Note that for natural numbers, we use the same symbol $\leq$ and denote by $(\mathbb{N}, \leq)$ the natural order and by $(\mathbb{N}^k, \leq)$ the product order.

Let $(X, \leq)$ be a po. For all $s_1, s_2 \in X$, we say that $s_3 \in X$ is the *supremum* (or sup) of s_1 and s_2 noted $sup(s_1, s_2)$ if s_3 is an upper bound of s_1 and s_2 (*i.e.* $s_1 \leq s_3$ and $s_2 \leq s_3$), and for all upper bounds $s_4 \in X$, we have $s_3 \leq s_4$.

Let $(X, \leq)$ be a qo. An *upward-closed* set is a subset $Y \subseteq X$ such that if $x \leq y$ and $x \in Y$ then $y \in Y$. For some $x \in X$, we write $\uparrow x = \{y \in X \mid x \leq y\}$ its upward-closure, and for some $Y \subseteq X$, $\uparrow Y = \bigcup_{x \in Y} \uparrow x$. A *basis* of an upward-closed subset $Y \subseteq X$ is any set B such that $Y = \uparrow B$. We say that an upward closed set Y has a *finite basis* if there exists some finite basis B of Y.

A qo $(X, \leq)$ is *well-founded* if there is no infinite sequence $(x_n)_{n \in \mathbb{N}}$ over X such that $x_{i+1} \leq x_i$ for every $i \in \mathbb{N}$. It is a *well-quasi-ordering* (wqo) if every infinite sequence $(x_n)_{n \in \mathbb{N}}$ over X contains an increasing pair, *i.e.* $\exists i < j$ such that $x_i \leq x_j$. If the wqo is a po, then it is called a *well partial order* (wpo). An *antichain* is a set in which each pair of different elements is incomparable.

Proposition 3 ([26])**.** *If $\leq$ is a qo on X then the following are equivalent:*

1. *$\leq$ is a wqo;*
2. *Every infinite sequence $(x_n)_{n \in \mathbb{N}}$ in X contains an infinite nondecreasing subsequence $x_{n_0} \leq x_{n_1} \leq ...$ with $n_0 < n_1 < ...$;*
3. *$\leq$ is well-founded and has no infinite antichain;*
4. *Every upward-closed subset $U \subseteq X$ has a finite basis;*
5. *Every nondecreasing sequence $U_0 \subseteq U_1 \subseteq ... \subseteq U_i \subseteq ...$ of upward-closed subsets U_i of X eventually stabilizes.*

1.3.2 Well Structured Transition Systems

The theory of *Well-Structured Transition Systems* [19, 1, 18] has been developed to unify and generalize some decidability results on termination, boundedness and coverability problems for infinite state systems equipped with a wqo on their states and a monotone transition relation with regards to this wqo. However, wqos are typically hard to find for complex systems. Sometimes, we will focus on transition systems with weaker constraints, that is, monotone transition systems.

We consider *Ordered Transition System* (OTS) $\mathcal{S} = (Q, \rightarrow, \leq)$ where $(Q, \rightarrow)$ is a transition system and Q is equipped with a quasi-ordering $\leq$. We are mainly interested here in the coverability problem for ordered transition systems, which consists, given an OTS $\mathcal{S} = (Q, \rightarrow, \leq)$, a set of initial states $I \subseteq Q$ and a state $s \in Q$, in deciding whether there exists a (possibly empty) run $i \xrightarrow{*} s'$ such that $s \leq s'$ and $i \in I$. We says that s is *coverable* if there is such a run. Let us denote the coverability problem by the predicate $cov(\mathcal{S}, I, s)$.

As shown in [1], the coverability problem can be solved by computing the set $Pred^*(\uparrow s)$ of all predecessors of $\uparrow s$ and then it suffices to check whether $I \cap Pred^*(\uparrow s)$ is empty to verify whether s is coverable. To compute $Pred^*(\uparrow s)$, we can iteratively compute the sequence of sets of predecessors $Pred(\uparrow s)$, $Pred^2(\uparrow s) = Pred(Pred(\uparrow s))$, ..., $Pred^n(\uparrow s)$, ... and we have $Pred^*(\uparrow s) = \bigcup_{n \geq 0} Pred^n(\uparrow s)$. But this sequence, in general, does not necessarily stabilize.

For monotone (ordered) transition systems, the set $Pred^*(\uparrow s)$ is upward closed; and if moreover $\leq$ is a wqo, the set $Pred^*(\uparrow s)$ will also have a finite basis. This opens the way to compute this finite basis which represents the set $Pred^*(\uparrow s)$.

Definition 5 (Monotone transition system [18]). *A monotone transition system (MTS) $\mathcal{S} = (Q, \rightarrow, \leq)$ is an ordered transition system such that for all $q_1 \leq q_1'$ and transition $q_1 \rightarrow q_2$, there exists a run $q_1' \xrightarrow{*} q_2'$ such that $q_2 \leq q_2'$*

In Proposition 3.1 of [18], the monotony of an ordered transition system is presented as the compatibility of $\leq$ with the transition relation $\rightarrow$. Now if a monotone transition system $\mathcal{S} = (Q, \rightarrow, \leq)$ has a wqo $\leq$, it is a WSTS.

Definition 6 (Well-Structured Transition System [18]). *A* Well-Structured Transition System *$\mathcal{S} = (Q, \rightarrow, \leq)$ is a monotone transition system such that $\leq$ is a wqo on Q.*

For example, Petri Nets are WSTSs if we take the inclusion of markings as the partial order. Indeed, the multiset inclusion is wpo and the wpo is compatible with the transition relation. Lossy Channel Systems [20] are another example of WSTS.

Monotone transition systems and WSTSs are both supposed to be *effective*: $\leq$ is decidable (*i.e.* there exists an algorithm determining whether a pair of elements belongs to $\leq$) and $\rightarrow$ is decidable.

The following lemma relies on the monotony condition.

Lemma 1 ([18]). *For a monotone transition system $\mathcal{S} = (Q, \rightarrow, \leq)$ and $q \in Q$, we have $\uparrow Pred^*(\uparrow q) = Pred^*(\uparrow q)$.*

To make the iterative computation of the $Pred^n(\uparrow q)$, we need an OTS with *effective pred-basis.*

Definition 7 (Effective pred-basis [18]). *An OTS $\mathcal{S} = (Q, \rightarrow, \leq)$ has* effective pred-basis *if there exists an algorithm accepting any state $q \in Q$ and returning $pb(q)$, a finite basis of $\uparrow Pred(\uparrow q)$.*

For an OTS $\mathcal{S} = (Q, \rightarrow, \leq)$ with effective pred-basis, and a finite subset of states $F \subseteq Q$, the backward coverability analysis consists in computing a sequence of finite sets of states $K_0, K_1, ..$ such that $K_0 = F$ and $K_{n+1} = K_n \cup pb(K_n)$. We may verify that $Pred^*(\uparrow F) = \bigcup_{i \geq 0} \uparrow K_i$.

Lemma 2 ([18]). *Let $\mathcal{S} = (Q, \rightarrow, \leq)$ be a monotone transition system and $s \in Q$. If there exists $m \in \mathbb{N}$ such that $\uparrow K_m = \uparrow K_{m+1}$, then $\uparrow K_m = Pred^*(\uparrow s)$.*

Effective pred-basis condition is sufficient to show that a finite basis for each $\uparrow K_i$ is computable according to the definition of the K_i. Since we suppose that MTSs and WSTSs are effective, the qo $\leq$ is decidable, and then inclusion $\uparrow K_n \supseteq \uparrow K_{n+1}$ and equality $\uparrow K_n = \uparrow K_{n+1}$ can be tested. Moreover, if $\leq$ is a wqo, then $(\uparrow K_i)_{i \in \mathbb{N}}$ converges. Hence, the iterative computation of K_n gives a procedure to the problem of reachability of $\uparrow s$ (*i.e.*, the coverability of s).

Theorem 1 ([18]). *For a WSTS $\mathcal{S} = (Q, \rightarrow, \leq)$ with effective pred-basis and $s \in Q$, a finite basis $B \subseteq Q$ of $Pred^*(\uparrow s)$ is computable. Hence, for any set of initial states $I \subseteq Q$, $cov(\mathcal{S}, I, s)$ is decidable, assuming the emptiness of $I \cap \uparrow B$ is decidable.*

Remark that if I is a finite set or an upward-closed set given by a finite basis, checking the emptiness of $I \cap \uparrow B$ is straightforward with a decidable $\leq$. But here we consider any I (because in PASTD the set of initial states is infinite and not upward-closed) and state the decidability of $I \cap \uparrow B = \emptyset$ as a condition compared to [18].

1.3.3 Monotone PASTD

In this section, we present a qo $\preceq$ based on the subgraph relation such that PASTDs are monotone.

To apply the theory of WSTSs to PASTDs, we need to define a qo on the set of states. But first, let us define an equivalence relation on states. We consider, from a reachability and coverability perspective, that the identifier of an entity instance has no importance and can always be replaced by another one; this has the advantage of reducing the computational complexity while allowing for a qo to be defined for a subset of PASTD. We consider that two states are equivalent if they are syntactically equivalent after renaming the entity identifiers using a permutation.

We call *permutation* any bijection $f : X \rightarrow X$. A *graph isomorphism* from (V, E) to (V', E') is a bijective function $f : V \rightarrow V'$ such that for all $v_1, v_2 \in V$, $\{v_1, v_2\} \in E$ iff $\{f(v_1), f(v_2)\} \in E'$.

Definition 8 (State equivalence). *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD and $s, s' \in \mathcal{T}_A$ such that $s = (V, E, \lambda)$ and $s' = (V', E', \lambda')$. We define $s \sim s'$ if there is a graph isomorphism ϕ from (V, E) to (V', E') such that for each P_i there is a permutation σ_i on P_i such that for all $v \in V$, $\sigma_i(\lambda(v)) = \lambda'(\phi(v))$ if $\lambda(v) \in P_i$.*

Recall that for a PASTD $A = (F, \overrightarrow{T}, \overrightarrow{P})$, the set of labels for the nodes of the tree representation of a state is given by the set $\{(\mathsf{aut}_\circ, q_1), ..., (\mathsf{aut}_\circ, q_j), ||_\circ, |{:}_\circ, |||{:}_\circ\} \cup \bigcup_i P_i$. Definition 8 means that the the entity instance identifiers are irrelevant for our purpose. Hence, we can always rename an identifier of P_i by another one of the same P_i. For example, the state of Figure 1.3 is equivalent to the state of Figure 1.8 where the book b_1 is borrowed by the member m_2. Take the permutations σ_b and σ_m on P_b and P_m respectively such that $\sigma_b = id_{P_b} \oplus \{b_1 \mapsto b_2, b_2 \mapsto b_1\}$ and $\sigma_m = id_{P_m} \oplus \{m_1 \mapsto m_2, m_2 \mapsto m_1\}$, where id_X the identity on the set X and $f \oplus g$ the overriding of $f : X \rightarrow Y$ by $g : W \rightarrow Y$, *i.e.* $(f \oplus g)(x) = g(x)$ if $x \in W$ and $(f \oplus g)(x) = f(x)$ otherwise.

To define a quasi-ordering on the set of states, we use the definition of induced subgraph. (V, E) is called *induced subgraph* of (V', E') if $V \subseteq V'$ and $E = \{\{v_1, v_2\} \in E' \mid v_1, v_2 \in V\}$. Let $(\Lambda, \leq)$ be a qo set of labels. We define by $\sqsubseteq$ the extension of the induced subgraph relation (modulo isomorphism) to labeled graphs such that $(V, E, \lambda) \sqsubseteq (V', E', \lambda')$ if there exists f an isomorphism from (V, E) to an induced subgraph of (V', E') such that $\lambda(v) \leq \lambda'(f(v))$ for all $v \in V$. If no ordering on Λ is specified, we assume that $\sqsubseteq$ is defined according to the identity (Λ, id_Λ).

Definition 9 (State quasi-ordering). *Let A be a PASTD and $s, s' \in \mathcal{T}_A$. We define $s \preceq s'$ if there exists $s'' \sim s$ such that $s'' \sqsubseteq s'$.*

As we consider only well-formed states for a specific PASTD, subtree states can only be obtained by pruning branches from a quantified interleaving node.

Proposition 4. *PASTDs under $\preceq$ are monotone transition systems.*

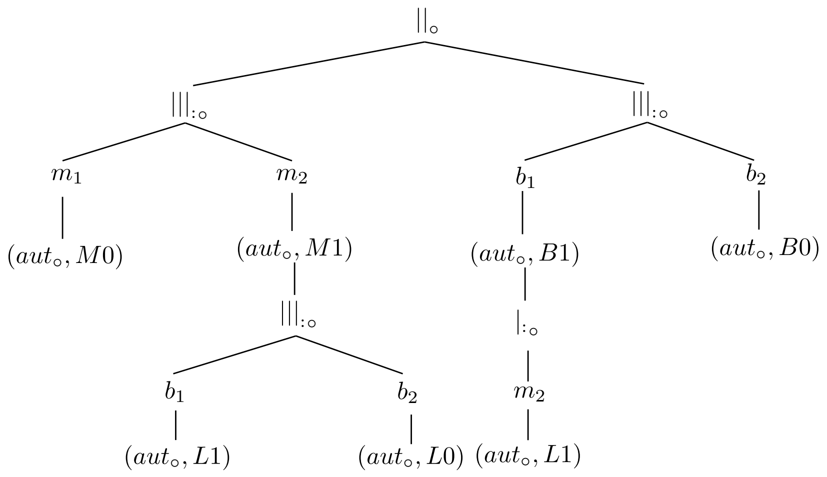


Figure 1.8: A state of the library system

Proof. The proof of monotony is similar to the one given in Appendix A for a superset of PASTD. □

1.3.4 Bounded-PASTD

PASTDs under $\preceq$ are not WSTSs because in general $\preceq$ is not wqo.

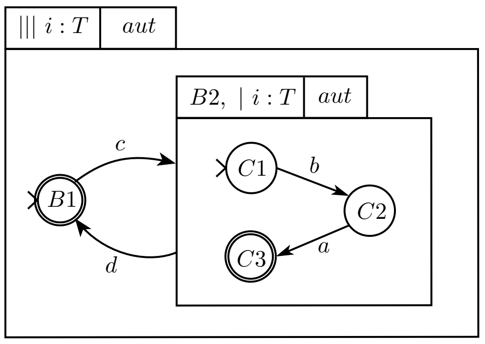
Proposition 5. *There is a PASTD such that $\preceq$ is not wqo.*

Proof. See Figure 1.9 for an example of infinite antichain. □

We have not found yet a useful wqo for all PASTDs. Nonetheless, for some ASTDs, the set of states is wqo. For example, it is obviously the case for ASTDs without parameters, as the set of states is then finite. Some other weaker conditions can be used to get a wqo. We will define in the following a class of PASTDs satisfying the wqo property. But first, we propose another representation of the state. Indeed, we aim at using a wqo theorem from [13] (see Theorem 2 in the sequel) and we need a graph structure with a wqo on the set of labels. However, we did not find any adequate wqo on the set of label if we keep the same tree structure (that is also why we cannot use Kruskal's wqo theorem on trees [31]). Hence, we use the following representation, where we replace labels denoting values e of P_i by new nodes which are labeled by i, thus abstracting from the particular value of e. This little trick removes infinite sequences of incomparable elements without losing critical information.

Definition 10 (Graph of s)**.** *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD. For every $s = (V, E, \lambda) \in \mathcal{T}_A$ with $(R_1, ..., R_n) = val(s)$, let $g(s) = (V', E', \lambda')$ be defined as follows:*

- $V' = V \cup \{(i, p) \mid i \in 1..n \wedge p \in R_i\}$
- $E' = E \cup \{\{(i, p), w\} \mid i \in 1..n \wedge p \in R_i \wedge w \in V \wedge \lambda(w) = p\}$
- $\lambda' = \lambda \oplus \{(i, p) \mapsto i \mid i \in 1..n \wedge p \in R_i\} \oplus \{w \mapsto 0 \mid \exists i \in 1..n \cdot \lambda(w) \in R_i\}$



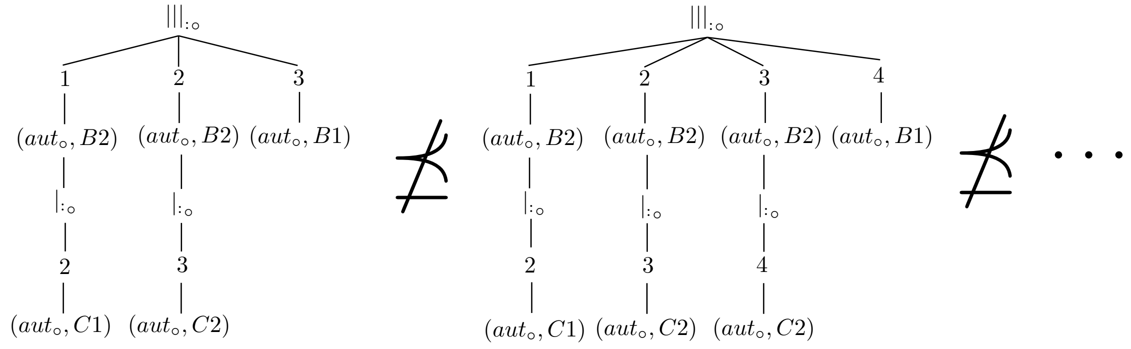


Figure 1.9: An infinite antichain in $(\mathcal{T}_A, \preceq)$

Intuitively, the function g consists in replacing labels p from parameter sets R_i by using new nodes of the form (i,p) and new edges between new nodes and nodes labeled from R_i. Each new node (i,p) will be connected with each node that has the same label in P_i. Old labels p from R_i are replaced with a default value 0. New nodes (i,p) are labeled with i. In this way, we obtain another representation of the state, which is still a graph but not a tree. But the codomain of the labeling λ' for the set of all possible states $\mathcal{T}_A$ is now bounded as it does not include $\bigcup_i P_i$ anymore. Note that each equivalent state in the previous version has the same graph representation in the new version. An example is given in Figure 1.10 which shows the graph corresponding to the state of Figure 1.3. The red labels represent the new nodes and the blue ones are the old labels. The dotted lines are the new added edges.

We denote by $\mathcal{G}_A = g(\mathcal{T}_A)$ the set of new graph representations for states and by Λ_A the set of all labels in $\mathcal{G}_A$, *i.e.* the codomain of each labeling function.

$$\Lambda_A = \{(\mathsf{aut}_\circ, q_1), ..., (\mathsf{aut}_\circ, q_j), ||_\circ, |\!:_\circ, |||\!:_\circ\} \cup \bigcup_{i \in 1..n} \{i\} \cup \{0\}$$

We introduce a new class of PASTD called Bounded-PASTD which satisfies the wqo property. In a graph (V,E), a *path* is a sequence of vertices $v_1, v_2, ..., v_n \in V$ such that $\{v_i, v_{i+1}\} \in E$ for all $i \in 1..n-1$. A *simple path* is a path $v_1, v_2, ..., v_n \in V$ such that $v_i \neq v_j$ for all $i \neq j$. We denote by $\mathcal{P}_k(\Lambda)$ the set of graphs $\mathcal{P}_k$ whose vertices are labeled by the set Λ. Let $\mathcal{P}_k$ the class of graphs without simple paths of length k.

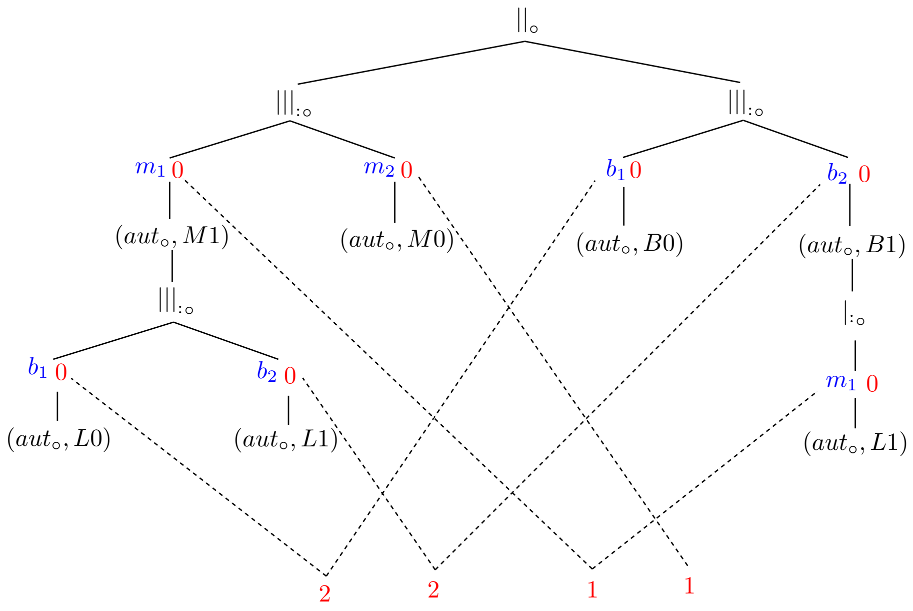


Figure 1.10: The graph representation of the state from Figure 1.3

Definition 11 (Bounded-PASTD)**.** *A PASTD A is a Bounded-PASTD if there exists $k \in \mathbb{N}$ such that $\mathcal{G}_A \subseteq \mathcal{P}_k(\Lambda_A)$.*

Recall that $\sqsubseteq$ is the induced subgraph relation for labeled graphs with regards to a qo set of labels $(\Lambda, \leq)$. When $\mathcal{G}_A$ is only composed of graphs whose simple paths are bounded, we can refer to Ding's wqo theorem [13] to show that $(\mathcal{G}_A, \sqsubseteq)$ is a wqo, and thus to prove that Bounded-PASTDs are well-structured.

Theorem 2 (Ding's theorem [13])**.** *For $k \in \mathbb{N}$, $(\mathcal{P}_k(\Lambda), \sqsubseteq)$ is a wqo if $(\Lambda, \leq)$ is a wqo.*

We denote by $\mathcal{T}_A/\!\sim$ the quotient set of $\mathcal{T}_A$ with regards to the equivalence relation $\sim$ and we use the same symbol $\preceq$ to denote the corresponding partial order between the equivalence classes. Let $\tilde{g} : \mathcal{T}_A/\!\sim \to \mathcal{G}_A$ the function such that $\tilde{g}(\tilde{s}) = g(s)$ for all $s \in \mathcal{T}_A$, where $\tilde{s}$ denotes the equivalence class of s. Consider the qo $(\mathcal{G}_A, \sqsubseteq)$ using the identity $(\Lambda_A, id_{\Lambda_A})$ as the qo on the set of labels. Similarly to graph isomorphisms, we call isomorphism from a qo $(X, \leq_X)$ to a qo $(Y, \leq_Y)$ any bijective function $f : X \to Y$ such that for all $s_1, s_2 \in X$, $s_1 \leq_X s_2$ iff $f(s_1) \leq_Y f(s_2)$. Such isomorphisms preserve wqo.

Lemma 3. *For any PASTD A, $\tilde{g}$ is an isomorphism from $(\mathcal{T}_A/\!\sim, \preceq)$ to $(\mathcal{G}_A, \sqsubseteq)$.*

Proof. Let $A = (F, \{T_1, ..., T_n\}, \{P_1, ..., P_n\})$. Since $\mathcal{G}_A$ is the image set of g, *i.e.* $\mathcal{G}_A = g(\mathcal{T}_A)$, we have that g is surjective. Thus, $\tilde{g}$ is also surjective. Let us prove that there exists $f : \mathcal{G}_A \to \mathcal{T}_A/\!\sim$ such that $f \circ \tilde{g}$ is the identity on $\mathcal{T}_A/\!\sim$. Let $t = (V, E, \lambda) \in \mathcal{G}_A$. We define $f(t)$ as the equivalence class of the state $s = (V', E', \lambda')$ which is constructed as follows. For all $i \in 1..n$, let $V_i = \{v \in V \mid \lambda(v) = i\}$ and $\lambda_i : V_i \to P_i$ an arbitrary injection. Let $V' = V \setminus \bigcup_{i \in 1..n} V_i$ and $E' = E \setminus \{\{v, w\} \in E \mid \exists i \in 1..n \cdot w \in V_i\}$ and $\lambda' = (\lambda \oplus \bigcup_{i \in 1..n, w \in V_i} \{v \mapsto \lambda_i(w) \mid \{v, w\} \in E\})\restriction_{V'}$, where $f\restriction_X$ is the domain restriction of the function f to X. It is easy to see that $f(\tilde{g}(\tilde{s})) = \tilde{s}$ for $\tilde{s} \in \mathcal{T}_A/\!\sim$ by definition of $\tilde{g}$ and f. Hence, the function $\tilde{g}$ is also injective. Let $x, y \in \mathcal{T}_A/\!\sim$. If

$x \preceq y$ then there are $s_1 \in x$, $s_2 \in y$ such that $s_1 \sqsubseteq s_2$. By definition of g, $g(s_1) \sqsubseteq g(s_2)$. By the definition of $\tilde{g}$, $\tilde{g}(x) = g(s_1)$ and $\tilde{g}(y) = g(s_2)$. Hence $\tilde{g}(x) \sqsubseteq \tilde{g}(y)$. Now suppose that $\tilde{g}(x) \sqsubseteq \tilde{g}(y)$. Then, for all $s_1 \in x$ and $s_2 \in y$, $g(s_1) \sqsubseteq g(s_2)$. Then, take $s_1 \in x$, $s_2 \in y$ such that $s_1 \preceq s_2$. Hence, $x \preceq y$. □

Theorem 3. *For any Bounded-PASTD A, $(\mathcal{T}_A, \preceq)$ is wqo.*

Proof. Let $k \in \mathbb{N}$ satisfying Definition 11. By Ding's theorem $(\mathcal{P}_k(\Lambda_A), \sqsubseteq)$ is wqo. Then, any subset $\mathcal{G}_A \subseteq \mathcal{P}_k(\Lambda_A)$ is wqo and by Lemma 3 $(\mathcal{T}_A/\!\sim, \preceq)$ is wqo, because isomorphisms preserve wqo. Thus, $(\mathcal{T}_A, \preceq)$ is wqo. □

As Bounded-PASTDs are also monotone, we can conclude the following result.

Proposition 6. *Bounded-PASTDs are WSTSs.*

We will show in Section 1.4.2 that the coverability problem is decidable for Bounded-PASTDs.

1.3.5 Classes of Bounded-PASTD

As deciding if a PASTD is a Bounded-PASTD may not be easy, in the following we propose two subclasses of Bounded-PASTD. For instance, a PASTD $A = (F, \overrightarrow{T}, \overrightarrow{P})$ where each parameter T_i appears only once in the expression F is called a 1-PASTD and a PASTD without nested quantifications is a Flat-PASTD. From an IS point of view, a 1-PASTD represents a system in which the entities can only be associated by weak entity relationships and a Flat-PASTD a system without relationships. Let us define formally these subclasses of PASTD.

As illustrated in Figure 1.2, recall that for a PASTD $A = (F, \overrightarrow{T}, \overrightarrow{P})$, where F is given in its tree notation (V, E, λ), the codomain of λ is the set of labels $\Lambda = \bigcup_i (\{\mathcal{A}_i\} \cup \bigcup_j \{q_{i,j}\}) \cup \bigcup_i \{||_{\Delta_i}\} \cup \bigcup_{i \in 1..n}\{\,|_{x \in T_i}\} \cup \bigcup_{i \in 1..n}\{\,|||_{x \in T_i}\}$, where the $\mathcal{A}_i$ are the different automaton ASTDs, $q_{i,j}$ the states of $\mathcal{A}_i$ and $||_{\Delta_i}$ the synchronization ASTDs.

Definition 12 (1-PASTD)**.** *Let $A = (F, \{T_1, ..., T_n\}, \{P_1, ..., P_n\})$ be a PASTD with $F = (V, E, \lambda)$. We call A a 1-PASTD if for all $i \in 1..n$, the set of nodes $\{v \in V \mid \lambda(v) \in \{\,|_{x \in T_i}, |||_{x \in T_i}\}\}$ is a singleton set.*

In a (labeled acyclic connected) graph (V, E, λ) that is also a tree, we denote by $ch(v)$ the child nodes of v and by $des(v) = ch^+(v)$ its transitive closure, *i.e.* the sets of descendants of v. Similarly, we denote by $pa(v)$ and $anc(v) = pa^+(v)$ the parent and ancestors of v respectively.

Definition 13 (Flat-PASTD)**.** *Let $A = (F, \{T_1, ..., T_n\}, \{P_1, ..., P_n\})$ be a PASTD with $F = (V, E, \lambda)$. Let $V_q = \{v \in V \mid \lambda(v) \in \bigcup_{i \in 1..n}\{\,|_{x \in T_i}, |||_{x \in T_i}\}\}$ (the quantified operator nodes). We call A a Flat-PASTD if for all $v \in V_q$ and all $w \in des(v)$, $\lambda(w) \notin \bigcup_{i \in 1..n}\{\,|_{x \in T_i}, |||_{x \in T_i}\}$.*

For instance, the PASTD simulating a VASS in Figure 1.7 is a Flat-PASTD as there is no nested quantified interleaving operators. It is also a 1-PASTD because T_1 and T_2 appear only in one quantified interleaving operator each.

Proposition 7. *Any 1-PASTD is a Bounded-PASTD.*

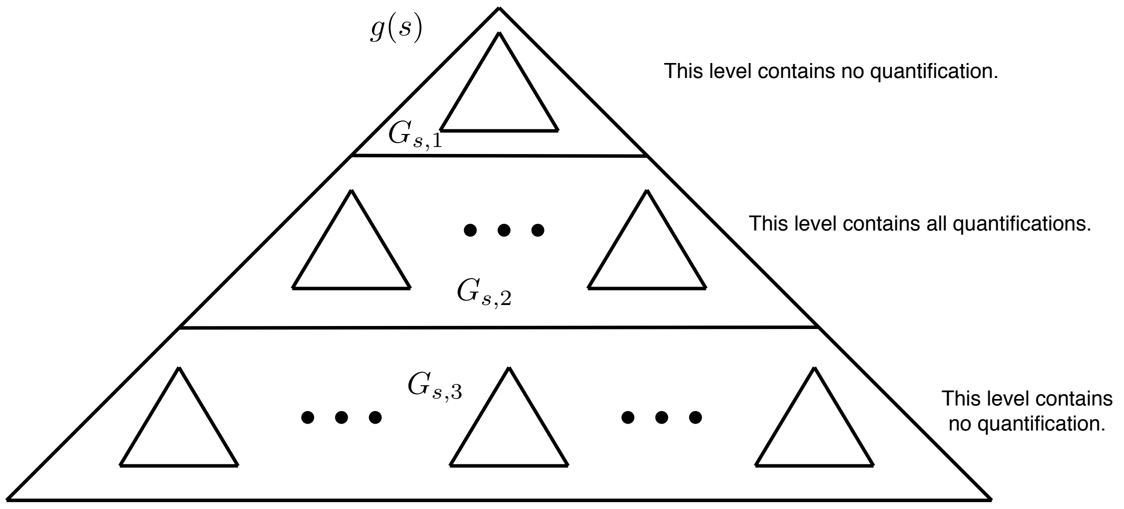

$g(s)$
This level contains no quantification.
$G_{s,1}$
This level contains all quantifications.
$G_{s,2}$
$G_{s,3}$
This level contains no quantification.

Figure 1.11: Node partition of a state

Proof. Let $s \in \mathcal{T}_A$ such that $s = (V, E, \lambda)$ and $g(s) = (V', E', \lambda')$. It is easy to see that $g(s)$ is a tree because every new node $v \in V' \setminus V$, which represents a value of a quantified node, is connected to only one other node. Moreover, the depth of s is bounded by the depth of F. Thus, every simple path in $\mathcal{G}_A$ is bounded. □

Now, let us prove that Flat-PASTDs are Bounded-PASTDs. Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ a PASTD, $s = (V, E, \lambda) \in \mathcal{T}_A$ with $(R_1, ..., R_n) = val(s)$ and $g(s) = (V', E', \lambda')$. V can be decomposed into three disjoint sets $V = V_{s,1} \cup V_{s,2} \cup V_{s,3}$ as follows:

- $V_{s,1} = \{v \in V \mid \lambda(v) \notin \{|:_\circ, |||:_\circ\} \wedge \lambda(anc(v)) \cap \{|:_\circ, |||:_\circ\} = \emptyset\}$, the nodes of the subtree containing the root of s and all descendants until the first quantified operator;
- $V_{s,2} = \{v \in V \mid \lambda(v) \in \{|:_\circ, |||:_\circ\} \cup \bigcup_i P_i \vee$
  $(\lambda(anc(v)) \cap \{|:_\circ, |||:_\circ\} \neq \emptyset \wedge \lambda(des(v)) \cap \bigcup_i P_i \neq \emptyset)\}$,

  the nodes of subtrees whose root is a quantified operator and including all descendants down to the quantification values;
- $V_{s,3} = V \setminus (V_{s,1} \cup V_{s,2})$, the remaining nodes.

Intuitively, as $V_{s,2}$ is the only part containing values of quantification, then only the nodes of $V_{s,2}$ have a different label through g in V'. V' can be decomposed into three disjoint sets $V'_{s,1} = V_{s,1}$, $V'_{s,2} = V_{s,2} \cup \bigcup_{i \in 1..n, p \in R_i} \{(i, p)\}$ and $V'_{s,3} = V_{s,3}$. We denote by $G_{s,1}$, $G_{s,2}$ and $G_{s,3}$ the subgraphs of s induced by $V_{s,1}$, $V_{s,2}$ and $V_{s,3}$ respectively and $G'_{s,2}$ the subgraph of $g(s)$ induced by $V'_{s,2}$. See Figure 1.11.

Lemma 4. *Let A be a PASTD. There exists $k \in \mathbb{N}$ such that for all $s \in \mathcal{T}_A$, $G_{s,1}$ contains at most k nodes and $G_{s,3}$ is composed of trees whose depth is at most k.*

Proof. As there is no quantified operator in $V_{s,1}$, then by the definition of the state structure, the number of nodes is bounded and depends on F only. Same reasoning for each tree in $G_{s,3}$. □

Lemma 5. *Let A be a PASTD. A is a Bounded-PASTD iff there exists $k \in \mathbb{N}$ such that for all $s \in \mathcal{T}_A$, $G'_{s,2} \in \mathcal{P}_k(\Lambda_A)$.*

Proof. Direction $\Rightarrow$. By Definition 11, if A is a Bounded-PASTD, then there is $k \in \mathbb{N}$ such that for all $s \in \mathcal{T}_A$, we have $g(s) \in \mathcal{P}_k(\Lambda_A)$. Thus, $G'_{s,2} \in \mathcal{P}_k(\Lambda_A)$. Direction $\Leftarrow$. Let $k_1 \in \mathbb{N}$ such that for all $s \in \mathcal{T}_A$, we have $G'_{s,2} \in \mathcal{P}_{k_1}(\Lambda_A)$. Let $s = (V, E, \lambda) \in \mathcal{T}_A$. Let us prove that there is $k \in \mathbb{N}$ such that $g(s) \in \mathcal{P}_k(\Lambda_A)$. By Lemma 4, let $k_2 \in \mathbb{N}$ such that $G_{s,1}$ contains at most k_2 nodes and $G_{s,3}$ is composed of trees whose depth is at most k_2. Let $u = (v_1, ..., v_m)$ be a simple path in $g(s)$ with $v_i \in V$ for all $i \in 1..m$ and $m \in \mathbb{N}$ its size. If u crosses only $G_{s,1}$, then $m \leq k_2$. If u crosses only $G_{s,3}$, then $m \leq 2 \times k_2$, as there are only trees of depth $\leq k_2$ in $G_{s,3}$. If u crosses only $G'_{s,2}$, then $m \leq k_1$. A more general simple path u in $g(s)$ can be decomposed into several segments $u_1, ..., u_j$ such that each segment crosses only one of the subgraphs $G_{s,1}$, $G'_{s,2}$ or $G_{s,3}$. Typically, for a path having the most segments, remark that the first segment begins in $G_{s,3}$, then the next ones alternate between $G'_{s,2}$ and (possibly) $G_{s,1}$ and the final one ends back in $G_{s,3}$. The path can only alternate k_2 times between $G'_{s,2}$ and $G_{s,1}$ as $G_{s,1}$ contains at most k_2 nodes. Thus, $m \leq 2 \times k_2 + k_1 + k_2 \times (k_2 + k_1) + 2 \times k_2 = k_2(4 + k_1 + k_2) + k_1$. Let $k' = k_2(4 + k_1 + k_2) + k_1$. We have $g(s) \in \mathcal{P}_{k'}(\Lambda_A)$. As k' is independent from s, $g(s) \in \mathcal{P}_{k'}(\Lambda_A)$ for all $s \in \mathcal{T}_A$. □

Proposition 8. *Any Flat-PASTD is a Bounded-PASTD.*

Proof. Let $A = (F, \{T_1, ..., T_n\}, \{P_1, ..., P_n\})$ a PASTD with $F = (V_F, E_F, \lambda_F)$. Let us prove that there is $k \in \mathbb{N}$ such that for all $s \in \mathcal{T}_A$, we have $G'_{s,2} \in \mathcal{P}_k(\Lambda_A)$, which is equivalent to being a Bounded-PASTD thanks to Lemma 5. Let $s \in \mathcal{T}_A$ and $g(s) = (V, E, \lambda)$. Because A has no nested quantifications, there are only two types of nodes in $V'_{s,2} = W_1 \cup W_2$, where $W_1 = \{v \in V \mid \lambda(v) \in \{|:_\circ, |||:_\circ\}\}$ the set of "quantified operator" nodes, and $W_2 = \{v \in V \mid \lambda(v) \in \{0, 1, ..., n\}\}$ the set of "parameters" nodes. Clearly, the longest simple path including only nodes from W_2 is of length 3, by construction of $g(s)$. Moreover, remark that, without nested quantifications, the size of W_1 is bounded by a number k_1 that depends on the structure F only (W_1 contains at most one $v \in V$ with $\lambda(v) \in \{|:_\circ, |||:_\circ\}$ for each $v_F \in V_F$ such that $\lambda_F(v_F) \in \bigcup_{i \in 1..n}\{\,|_{x \in T_i}, |||_{x \in T_i}\}$). By a similar reasoning as in the previous lemma, a "maximal" simple path alternates between W_1 and W_2 and its length is bounded by $k' = (1 + 3) \times k_1 = 4 \times k_1$. As k does not depend on s, $G'_{s,2} \in \mathcal{P}_{k'}(\Lambda_A)$ for all $s \in \mathcal{T}_A$. □

As 1-PASTDs and Flat-PASTDs are Bounded-PASTDs, we can conclude that we have two easily recognizable classes of PASTD that are WSTSs.

1.4 Computation of pred-basis for PASTDs

To apply the backward analysis algorithm presented in Section 1.3.2 to PASTDs, we need to prove a last condition that is the effective pred-basis. By definition, the existence of pred-basis condition can be related to the wqo condition, because with a wqo any upward-closed set has a finite basis. However, we think that these two conditions should be independent from each other to have a better understanding of the algorithm. Thus, we propose in Section 1.4.1 a new framework which identifies a set of conditions proving the effective pred-basis. Then, we show in Section 1.4.2 how to use the framework on PASTDs, hence to solve the coverability problem.

$$\begin{tikzcd}
 & s_1 \arrow[r] \arrow[d, phantom, sloped, "\geq"] & s_2 \arrow[d, phantom, sloped, "\geq"] & \\
\exists & q'_1 \arrow[r] \arrow[d, phantom, sloped, "\geq"] & q'_2 \arrow[d, phantom, sloped, "\geq"] & \forall \\
 & q_1 \arrow[r] & q_2 &
\end{tikzcd}$$

Figure 1.12: Condition 4 of RMTS : the backward-downward monotony

1.4.1 Ranked Monotone Transition Systems

In this section, we describe a general method to prove the *effective pred-basis* in some infinite systems without the wqo condition. We will show first that without wqo and under some other conditions, there exists a finite basis for $\uparrow Pred(\uparrow s)$. Then, we show that the finite pred-basis is computable with some effectivity hypothesis.

In order to compute a pred-basis for s, a naive approach would be to compute the upward closure of s and then the set of all predecessors. However, this procedure does not terminate when $\uparrow s$ is infinite. In practice, for most interesting systems, there is no need to consider the entire set of upper states to compute a basis of predecessors. Thus, we propose a method that determines which finite subset of $\uparrow s$ is sufficient to compute a finite basis of $\uparrow Pred(\uparrow s)$. To this end, we define a new class of transition systems called *Ranked Monotone Transition Systems.* Note that, to keep the definitions simple, we will consider OTSs with partial ordering in the following. We will see in the next section how to apply the method to OTSs with quasi-ordering.

Definition 14 (Ranked MTS)**.** *A Ranked Monotone Transition System $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ is a monotone transition system $(Q, \rightarrow, \leq)$, where $\leq$ is a po, with a function $\gamma : Q \rightarrow \mathbb{N}^n$ and $c, d \in \mathbb{N}^n$ s.t. :*

1. *γ is a monotone function and $\gamma^{-1}(r)$ is finite for all $r \in \mathbb{N}^n$,*
2. *for all $s_1, s_2, s_3 \in Q$ such that $s_1 \leq s_3$ and $s_2 \leq s_3$, there exists $s_4 \in Q$ such that $s_1 \leq s_4$, $s_2 \leq s_4$, $s_4 \leq s_3$ and $\gamma(s_4) \leq \gamma(s_1) + \gamma(s_2)$,*
3. *for all $s_1 \rightarrow s_2$, there exists $s'_1 \in Q$ such that $s'_1 \leq s_1$ and $s'_1 \rightarrow s_2$ and $\gamma(s'_1) \leq \gamma(s_2) + d$,*
4. *$\mathcal{S}$ is* backward-downward monotone *(bdm): for each transition $s_1 \rightarrow s_2$, there is a transition $q_1 \rightarrow q_2$ s.t.:*
   (a) $q_2 \leq s_2$, $\gamma(q_2) \leq c$, and
   (b) $\forall q'_2 \in Q \cdot q_2 \leq q'_2 \leq s_2 \implies \exists q'_1 \in Q \cdot q'_1 \rightarrow q'_2 \wedge q_1 \leq q'_1 \leq s_1$.

Definition 15 (Effective RMTS)**.** *An RMTS $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ is* effective *if $\rightarrow$ and $\leq$ are decidable and $\gamma(q)$ and $\gamma^{-1}(r)$ are both computable for all $(q, r) \in Q \times \mathbb{N}^n$.*

Let us give the intuition for each condition of Definition 14:

1. The function γ associates a vector of natural numbers called *rank* to each state of the system. Intuitively, the rank of a state represents its size and this size can be multidimensional. We can see the rank function as a way to split the partial ordering into several finite layers of states, each layer corresponding to a rank. This principle applies well to some ordered transition

systems. In particular, parameterized systems can be naturally equipped with rank functions. Indeed, a rank of a state may correspond to the number of instances of each type of entity in the state.

2. Condition 2 demands that if s_1 and s_2 have an upper bound s_3, then we can find a lesser one s_4 whose rank is bounded by the sum of the ranks of the two considered states.

3. To understand Condition 3, consider a parameterized system and suppose that the transition $s_1 \to s_2$ is "destructive", that is some instances of entities are lost during the transition and $\gamma(s_2) < \gamma(s_1)$. Then, the minimum number of instances that can disappear during this kind of transition must be lesser than d. Remark that if the transition relation never decreases the rank of the state, then the property is always satisfied.

4. The central property is Condition 4 and states that the *kernel* of a transition can be identified and is small. This *kernel* is the set of entity instances that change state in a transition (*e.g.*, the book and member involved in a loan transition). For each transition we look for a smaller one that has the "same semantics" and such that there exists intermediary transitions. See Figure 1.12 for a graphical representation of the backward-downward monotony. The value c is an upper bound on the sizes of all minimal transitions in the system. Intuitively, for each transition we identify a smaller part that is essential to fire the transition while the rest remains stable. The definition ensures that the minimal transition is a good one by checking that every intermediary transition is also possible.

Example 1 (Petri Nets)**.** *Let $N = (P, T, F)$ be a Petri Net where P is a set of places, T a set of transitions and $F : (P \times T \cup T \times P) \to \mathbb{N}$ a multiset of arcs. A marking is a multiset of places,* i.e. *a mapping $M : P \to \mathbb{N}$. We denote by $\mathcal{S}_N = (Q, \to, \subseteq)$ the corresponding transition system with Q the set of markings and $\subseteq$ the marking inclusion defined by $M_1 \subseteq M_2$ if $M_1(p) \leq M_2(p)$ for all place p. Let $\gamma : Q \to \mathbb{N}$ be a function such that $\gamma(M) = \sum_{p \in P} M(p)$.*

1. *Clearly, γ is monotone,* i.e. *$M_1 \subseteq M_2 \implies \gamma(M_1) \leq \gamma(M_2)$ and $\gamma^{-1}(r)$ is finite for all $r \in \mathbb{N}$ because P is finite.*

2. *For all $M_1, M_2, M_3 \in Q$ such that M_3 is an upper bound of M_1 and M_2, $sup(M_1, M_2) \subseteq M_3$ and $\gamma(sup(M_1, M_2)) \leq \gamma(M_1) + \gamma(M_2)$, because $sup(M_1, M_2)$ denotes $max(M_1(p), M_2(p))$ for each place p.*

3. *Let $d \in \mathbb{N}$ be the maximum number of tokens needed to fire a transition, then for all transitions $M_1 \to M_2$, we have $\gamma(M_1) \leq \gamma(M_2) + d$.*

4. *Finally, take $c = max(F)$,* i.e. *the maximum token consumed or created by a transition. Then, the bdm condition is verified. Indeed, for all transition $M_1 \to M_2$ fired by $t \in T$, we can always find a smaller one $M'_1 \to M'_2$ where for all $p \in P$, $M'_1(p) = F(p, t)$ and $M'_2(p) = F(t, p)$ such that Condition 4 is verified.*

Thus, $(Q, \to, \subseteq, \gamma, c, d)$ is an RMTS.

Example 2 (Petri Net Extensions)**.** *Petri Nets with transfer arcs are RMTSs, considering the same function γ. Condition 3 is satisfied because the number of token lost during the transition is still bounded. The backward-downward monotony holds because a transfer arc is always possible for* *any smaller marking. The same holds for Petri Nets with reset arcs. Note that the number of token lost during a reset transition is not bounded but for any reset transition there is always a lesser transition that is bounded.*

Example 3 (Lossy Channel Systems)**.** *Consider a Lossy Channel System $\mathcal{S}_C$ with Q a set of control states, Σ an alphabet and $c_1, ..., c_n$ a set of FIFO channels. The system can send a message along a channel, receive one from a channel or lose one in a channel. Each message is represented by one letter of Σ. Hence, a transition can add or remove one letter in a channel at a time. We denote by $\leq$ the ordering between configurations from $Q \times \Sigma^{*n}$ such that $(q, w_1, ..., w_n) \leq (q', w'_1, ..., w'_n)$ if $q = q'$ and $w_i \sqsubseteq w'_i$ for all $i \leq n$, where $\sqsubseteq$ is the sub-word ordering. Take the function $\gamma : Q \times \Sigma^{*n} \to \mathbb{N}^n$ such that $\gamma(q, w_1, ..., w_n) = (|w_1|, ..., |w_n|)$.*

1. *γ is monotone and $\gamma^{-1}(r)$ is finite for all $r \in \mathbb{N}^n$ as Q and Σ are finite.*
2. *Since $|sup(w, w')| \leq |w| + |w'|$ for all $w, w' \in \Sigma^*$, we have $\gamma(sup(s, s')) \leq \gamma(s) + \gamma(s')$ for all $s, s' \in Q \times \Sigma^{*n}$. Thus, the supremum satisfies Condition 2.*
3. *The size of a word in a channel can only increase or decrease by one, hence for every transition $s \to s'$ we have $\gamma(s) \leq \gamma(s') + (1, ..., 1)$.*
4. *For each type of transition we have a minimal one satisfying the bdm condition. If it is a send action $(q, w_1, ..., w_i, ..., w_n) \to (q', w_1, ..., w_i.a, ..., w_n)$, then take the smaller transition $(q, \epsilon, ..., \epsilon) \to (q', \epsilon, ..., a, ..., \epsilon)$. If it is a receive action $(q, w_1, ..., a.w_i, ..., w_n) \to (q', w_1, ..., w_i, ..., w_n)$, then take $(q, \epsilon, ..., a, ..., \epsilon) \to (q', \epsilon, ..., \epsilon, ..., \epsilon)$. Finally, if it is a loss of message $(q, w_1, ..., a.w_i, ..., w_n) \to (q, w_1, ..., w_i, ..., w_n)$, then take $(q, \epsilon, ..., a, ..., \epsilon) \to (q, \epsilon, ..., \epsilon, ..., \epsilon)$. Condition 4 is then verified for $c = (1, ..., 1)$.*

Thus, $(Q \times \Sigma^{*n}, \to, \leq, \gamma, c, c)$ is an RMTS.

Nicely Sliceable WSTS (NSW)[6] share some similarities with RMTSs. They both use the same notion of partitioning of the state space and of downward monotony. However, they rely on different assumptions and some of their requirements are unnecessary for our purpose. In [6], the authors propose an algorithm to compute the set of all predecessors $Pred^*(\uparrow s)$ and use the NSW framework in order to prove its correctness. That differs from our goal which is to give a complete method to prove the existence of pred-basis, that is not guaranteed without the wqo condition, and to compute it. Petri Nets, Broadcast Protocols, Lossy Channel Systems and Context-free grammars are examples of NSWs [6].

Definition 16 ([6])**.** *A δ-NSW (Nicely Sliceable WSTS) is a WSTS $\mathcal{S} = (Q, \to, \leq)$ such that*

1. *$\leq$ is discrete over Q,* i.e.
   (a) *$\forall q \in Q, \exists k \in \mathbb{N}$ s.t. for any sequence $q_0 \leq ... \leq q_l = q$ we have $l \leq k$ and there is a weight function $w : Q \to \mathbb{N}$ that maps each $q \in Q$ to the minimum such k.*
   (b) *$\{q \in Q \mid w(q) = i\}$ is finite for each $i \in \mathbb{N}$*
2. *$\mathcal{S}$ is weight respecting,* i.e.
   (a) *for all $x, x', y \in Q$ such that $x \to x'$ and $x \leq y$, there exists $y' \in Q$ such that $y \to y'$ and $w(x') - w(x) = w(y') - w(y)$.*

*3. $\mathcal{S}$ is δ-*deflatable*, i.e.*

   (a) for $\delta \in \mathbb{N}$, if $x \to x'$ and $z \leq x'$, then there exists y, y' such that
   - *$y \leq x$ and $y \to y'$ and $z \leq y'$,*
   - *$w(y) \leq w(z) + \delta$ and $w(y') \leq w(z) + \delta$.*

More precisely, the main differences between NSWs and RMTSs are that NSWs are based on a wqo and require the "weight respecting" condition whereas RMTSs need a po. "Weight respecting" is necessary in [6] to prove their convergence theorem (Theorem 1). Nonetheless, the RMTS condition 1 is similar to the NSW definition of discrete system (without wqo) and the RMTS conditions 2, 3 and 4 to the NSW definition of deflatability (where $\delta = max(c, d)$). Even if the RMTS conditions may look more complex, we think that stating these conditions instead of the deflatability one is more useful for the following reasons.

- The backward-downward monotony gives a better intuition, because the search for the right c is guided by the notion of the "least transition" whose states have their ranks bounded by c. Thus, the RMTS definition targets the biggest "least transition" of the system.
- Proving the existential condition of the deflatability, may require an explicit construction of the states y and y', which is not trivial. We will see in the proof of Lemma 6 that the conditions 2 and 4 give a direct construction of the state y' (the state y is then chosen among the predecessors of y').
- RMTSs distinguish between the two values c and d (instead of δ). That allow for a better precision at the computation of the pred-basis, as shown in the following.

The objective is now to show that RMTSs allow for the determination of effective pred-basis easily without wqo. In order to find a finite basis for $\uparrow Pred(\uparrow s)$, the idea is to compute a finite subset $S \subset \uparrow s$ and a finite subset $P \subseteq Pred(S)$.

Definition 17. *Let $\mathcal{S} = (Q, \to, \leq, \gamma, c, d)$ be an RMTS. For all $e \in \mathbb{N}^n$ let us denote $Pred_e(s) = \{q \in Pred(s) \mid \gamma(q) \leq \gamma(s) + e\}$ and $\uparrow_e s = \{q \in \uparrow s \mid \gamma(q) \leq \gamma(s) + e\}$.*

$Pred_c(s)$ and $\uparrow_c s$ restricts the set of predecessors and the upward-closure of s to the states that will be interesting to compute the pred-basis.

Lemma 6. *Let $\mathcal{S} = (Q, \to, \leq, \gamma, c, d)$ be an RMTS and $s \in Q$. We have $\uparrow Pred(\uparrow s) = \uparrow Pred_d(\uparrow_c s)$.*

Proof. $\supseteq$ is obvious. To prove $\subseteq$, let us show that $Pred(\uparrow s) \subseteq \uparrow Pred_d(\uparrow_c s)$ (then $\uparrow Pred(\uparrow s) \subseteq \uparrow Pred_d(\uparrow_c s)$ by upward closure). Let $p' \in Pred(\uparrow s)$ and $s' \in Q$ such that $p \to s'$ and $s \leq s'$. We want to show that there exists $p \in Q$ such that $p \leq p' \wedge \exists s'' \in Q \cdot s \leq s'' \wedge p \to s'' \wedge \gamma(s'') \leq \gamma(s) + c \wedge \gamma(p) \leq \gamma(s'') + d$. By the bdm, take $q_1 \to q_2$ such that $q_2 \leq s'$, $\gamma(q_2) \leq c$, and $\forall q'_2 \in Q \cdot q_2 \leq q'_2 \leq s' \implies \exists q'_1 \in Q \cdot q'_1 \to q'_2 \wedge q_1 \leq q'_1 \leq p'$. Take s'' an upper bound of s and q_2 such that $s'' \leq s'$, satisfying Condition 2. We have $q_2 \leq s'' \leq s'$. Thus, by the bdm, there is $p'' \in Q$ such that $q_1 \leq p'' \leq p'$ and $p'' \to s''$. By Condition 3, take $p \in Q$ such that $p \leq p''$, $p \to s''$ and $\gamma(p) \leq \gamma(s'') + d$. Finally, by Condition 2, $\gamma(s'') \leq \gamma(s) + \gamma(q_2) \leq \gamma(s) + c$. A graphical representation of the proof is given in Figure 1.13. □

We construct a basis for $\uparrow Pred(\uparrow s)$ by computing the set $Pred_d(\uparrow_c s)$. Let us show that this set is finite.

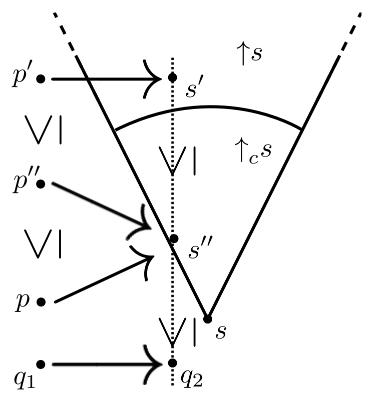


Figure 1.13: Illustration of Lemma 6

Lemma 7. *Let $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ an RMTS. For all $s \in Q$, $\uparrow Pred(\uparrow s)$ has a finite basis.*

Proof. $Pred_d(\uparrow_c s)$ is a basis for $\uparrow Pred(\uparrow s)$ by Lemma 6. The set $\{r \in \mathbb{N}^n \mid r \leq c\}$ is finite, thus $\uparrow_c s$ is finite by definition of a rank function. Similarly, for all $q \in Q$, $Pred_d(q)$ is finite. Consequently, $Pred_d(\uparrow_c s)$ is finite. □

Proposition 9. *Effective RMTSs have effective pred-basis.*

Proof. Let $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ be an effective RMTS. Let us show that for all $s \in Q$, $Pred_d(\uparrow_c s)$ is computable. As γ is effective, then for all $r \in \mathbb{N}^n$, $\gamma^{-1}(r)$ is computable. To compute $\uparrow_c s = \{q \in \uparrow s \mid \gamma(q) \leq \gamma(s) + c\}$, compute the finite set $X = \bigcup_{r \leq \gamma(s)+c} \gamma^{-1}(r)$ and select the elements $q \in X$ such that $s \leq q$. Then, for all $q \in \uparrow_c s$, compute $Pred_d(q) = \{q' \in Pred(q) \mid \gamma(q') \leq \gamma(q) + d\} = \{q' \in \bigcup_{r \leq \gamma(q)+d} \gamma^{-1}(r) \mid q' \rightarrow q\}$. □

This proof gives a naive algorithm to compute $Pred_d(\uparrow_c s)$, but in practice enumerating an entire set $\gamma^{-1}(r)$ is not always necessary. Especially, if $Pred(q)$ is computable, then computing $Pred_d(q)$ is more straightforward.

Example 4 (Petri Nets)**.** *Let $N = (P, T, F)$ be a Petri Net. We denote by $\mathcal{S}_N = (Q, \rightarrow, \subseteq, \gamma, c, d)$ the corresponding RMTS defined in Example 1 with $c, d \in \mathbb{N}$. Let M be a marking and let us compute a pred-basis of M. The set of markings $\uparrow_c M$ is clearly finite and can be enumerated. For each element M' of $\uparrow_c M$, $Pred(M')$ is also finite and computable, hence so is $Pred_d(M')$.*

Even if computing a pred-basis is easy for systems like Petri Nets or Lossy Channel Systems, finding one for some more complex system [30, 33] is more difficult. The RMTS framework gives a systematic way to prove the existence of pred-basis without the wqo hypothesis and gives a generic algorithm.

Now, if $\uparrow Pred(\uparrow s)$ is computable, then as shown is Section 1.3.2, we can compute $Pred^*(\uparrow s)$ by iteration assuming the number of iterations is finite. This is guaranteed by the fact that the quasi-ordering is a wqo.

Proposition 10. *For an effective RMTS $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ with $I \subseteq Q$ and $s \in Q$, if $\leq$ is a wqo, then a finite basis $B \subseteq Q$ of $Pred^*(\uparrow s)$ is computable. In addition, if $I \cap \uparrow B = \emptyset$ is decidable, then $cov(\mathcal{S}, I, s)$ is decidable.*

Proof. $\mathcal{S}$ has effective pred-basis thanks to Proposition 9. Since $\leq$ is a wqo, then $\mathcal{S}$ is a WSTS. Thus, by Theorem 1, the coverability problem is decidable. $\square$

Remark that if we do not have a wqo, we can still use the procedure computing $\mathit{Pred}^*(\uparrow s)$ of Section 1.3.2 as a semi-algorithm. Furthermore, like in [6], we have intermediate results by restricting ourselves to the verification for some specific ranks. Let $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ be an effective RMTS. Then, we can always consider a cut-off value $r \in \mathbb{N}^n$ and a subsystem $\mathcal{S}_r = (Q_r, \rightarrow, \leq)$ whose set of states is reduced to $Q_r = \{q \in Q \mid \gamma(q) \leq r\}$. During the computing of a base for $\mathit{Pred}^*(\uparrow s)$, we can conclude on the coverability of s in $\mathcal{S}_r$ as we go. Indeed, if all backward paths contain a state of rank r and no initial state is currently reached, then s is not coverable in $\mathcal{S}_r$. That is more efficient than choosing a value r and checking each $\mathcal{S}_r$ one by one. We just need to adapt the backward reachability procedure of Section 1.3.2 either by prioritizing lower ranks or by keeping track of computed paths.

1.4.2 Ranked PASTD

In this section we use RMTSs to prove that PASTDs have effective pred-basis. Because we use a quasi-ordering on the state space of PASTDs, we cannot say that PASTDs are RMTSs as it require a partial ordering. However, it is natural to consider the quotient set of the state space instead of the entire space when studying systems like PASTDs. Exploiting the symmetries in systems in order to simplify a verification process is usual in the context of formal verification [17]. In our case the symmetries entailed by the equivalence relation allow us to consider the coverability problem on the quotient space.

For a PASTD A, $\mathcal{T}_A/\sim$ is the quotient set of $\mathcal{T}_A$ and for any relation (transition relation or ordering in our case) on $\mathcal{T}_A$, we use the same notation for the corresponding one on $\mathcal{T}_A/\sim$. More formally, for all $x_1, x_2 \in \mathcal{T}_A/\sim$, $x_1 \preceq x_2$ iff $\exists s_1 \in x_1, \exists s_2 \in x_2 \cdot s_1 \preceq s_2$, and $x_1 \rightarrow x_2$ iff $\exists s_1 \in x_1, \exists s_2 \in x_2 \cdot s_1 \rightarrow s_2$. We denote by $\tilde{\mathcal{S}}_A = (\mathcal{T}_A/\sim, \rightarrow, \preceq)$ the quotient system.

Proposition 11. *Let A be a PASTD. Then, $(\mathcal{T}_A, \rightarrow, \sim)$ and $\tilde{\mathcal{S}}_A = (\mathcal{T}_A/\sim, \rightarrow, \preceq)$ are both monotone.*

Proof. The proof of monotony of $(\mathcal{T}_A, \rightarrow, \sim)$ is similar to the one for $(\mathcal{T}_A, \rightarrow, \preceq)$ (see Appendix A). Let us prove that $(\mathcal{T}_A/\sim, \rightarrow, \preceq)$ is monotone. Let $x_1, y_1, x_2 \in \mathcal{T}_A/\sim$ such that $x_1 \preceq y_1$ and $x_1 \rightarrow x_2$. Then, there exist $s_1, q_1 \in x_1$ and $s_2 \in x_2$ and $s'_1 \in y_1$ such that $q_1 \preceq s'_1$ and $s_1 \rightarrow s_2$. By monotony of $(\mathcal{T}_A, \rightarrow, \sim)$, there exists $q_2 \in x_2$ such that $q_1 \rightarrow q_2$. Thus, by monotony of $(\mathcal{T}_A, \rightarrow, \preceq)$, there exists $s'_2 \in \mathcal{T}_A$ such that $q_2 \preceq s'_2$ and $s'_1 \rightarrow s'_2$. Let y_2 be the equivalence class of s'_2. Then, we have $x_2 \preceq y_2$ and $y_1 \rightarrow y_2$. $\square$

The monotony of $(\mathcal{T}_A, \rightarrow, \sim)$ can be used to prove that the coverability problem in $\mathcal{S}_A$ can be reduced to the coverability problem in $\tilde{\mathcal{S}}_A$.

Theorem 4. *Let A be a PASTD, $\mathcal{S}_A = (\mathcal{T}_A, \rightarrow, \preceq)$ its corresponding OTS and $\tilde{\mathcal{S}}_A = (\mathcal{T}_A/\sim, \rightarrow, \preceq)$ its quotient OTS. Let $I \subseteq \mathcal{T}_A$, $s \in \mathcal{T}_A$, $\tilde{I} = \{\tilde{i} \mid i \in I\}$ and $\tilde{s}$ the equivalence class of s. Then, $cov(\mathcal{S}_A, I, s) \iff cov(\tilde{\mathcal{S}}_A, \tilde{I}, \tilde{s})$.*

Proof. Let us prove that $i \xrightarrow{*} s' \cdot s \preceq s' \wedge i \in I \Leftrightarrow \tilde{i} \xrightarrow{*} \tilde{s'} \cdot \tilde{s} \preceq \tilde{s'} \wedge \tilde{i} \in \tilde{I}$. $\Rightarrow$ is trivial. $\Leftarrow$ is true because $(\mathcal{T}_A, \rightarrow, \sim)$ is monotone (see Proposition 11). $\square$

From now on, for a PASTD A we will study the coverability problem on the quotient MTS $\tilde{\mathcal{S}}_A = (\mathcal{T}_A/\sim, \rightarrow, \preceq)$. The objective is to find an RMTS for A. Let us define an adequate rank function.

Definition 18. *For a PASTD $A = (F, (T_1, ..., T_n), (P_1, ..., P_n))$, let $\gamma : \mathcal{T}_A \rightarrow \mathbb{N}^n$ be the function defined by $\gamma(s) = (|R_1|, ..., |R_n|)$ where $\overrightarrow{R} = val(s)$ for all $s \in \mathcal{T}_A$.*

The function γ gives for each state s the number of instances of each type that appear within the description of s. For instance, if s is the state of Figure 1.3 from the library example, then $\gamma(s) = (2, 2)$ as $val(s) = (\{m_1, m_2\}, \{b_1, b_2\})$. The state describes a configuration with 2 members and 2 books.

Clearly, for all $s_1, s_2 \in x \in \mathcal{T}_A/\sim$, $\gamma(s_1) = \gamma(s_2)$. Thus, we can extend the rank function to the equivalence classes $\gamma : \mathcal{T}_A/\sim \rightarrow \mathbb{N}^n$ as $\gamma(\tilde{s}) = \gamma(s)$ for any $s \in \mathcal{T}_A$. The function has the following property with regards to transitions.

Proposition 12. *Let A be a PASTD. For all $x, y \in \mathcal{T}_A/\sim$, such that $x \rightarrow y$, there exists $z \preceq x$ such that $z \rightarrow y$ and $\gamma(z) = \gamma(y)$.*

Proof. By definition of the transition relation. See Appendix A. □

Now, let us prove that Condition 2 of Definition 14 is satisfied, *i.e.* that if two PASTD states have an upper bound, they have a "bounded" one according to the rank function γ.

Lemma 8. *Let A be a PASTD. For all $x_1, x_2, x_3 \in \mathcal{T}_A/\sim$ such that $x_1 \preceq x_3$ and $x_2 \preceq x_3$, there exists $x_4 \in \mathcal{T}_A/\sim$ such that $x_1 \preceq x_4$, $x_2 \preceq x_4$, $x_4 \preceq x_3$ and $\gamma(x_4) \leq \gamma(x_1) + \gamma(x_2)$.*

Proof. Let $A = (F, (T_1, ..., T_n), (P_1, ..., P_n))$. Let $x_1, x_2, x_3 \in \mathcal{T}_A/\sim$ such that $x_1 \preceq x_3$ and $x_2 \preceq x_3$. Then, there is $s_1 \in x_1, s_2 \in x_2, s_3 \in x_3$ such that $s_1 \preceq s_3$ and $s_2 \preceq s_3$. Let $s_1 = (V_1, E_1, \lambda_1)$, $s_2 = (V_2, E_2, \lambda_2)$ and $s_3 = (V_3, E_3, \lambda_3)$. By Definitions 8 and 9, as $s_1 \preceq s_3$ there exist an isomorphism f_{13} from (V_1, E_1) to an induced subgraph of (V_3, E_3) and permutations $\sigma_{13,i}$ on P_i for each $i \in 1..n$. Consider the same notations f_{23} and $\sigma_{23,i}$ for $s_2 \preceq s_3$. We construct $s_4 = (V_4, E_4, \lambda_4)$ as follows: $V_4 = \{v \in V_3 \mid v \in Im(f_{13}) \cup Im(f_{23})\}$, $E_4 = \{\{v, w\} \in E_3 \mid v, w \in V_4\}$ and $\lambda_4 = \lambda_3\restriction_{V_4}$. s_4 is a well-formed state because s_1, s_2 and s_3 are from the same PASTD and only branches from quantified interleaving nodes are pruned from s_3 to obtain s_4. Hence, $s_4 \in \mathcal{T}_A$. Clearly, $s_4 \preceq s_3$ by construction. Using the same injection f_{13} and permutations $\sigma_{13,i}$, we have that $s_1 \preceq s_4$. Same for $s_2 \preceq s_4$. By definition of γ, we have that $\gamma(s_4) \leq \gamma(s_1) + \gamma(s_2)$ because for each $v \in V_4$ with $\lambda_4(v) \in P_i$ either there is $w \in V_1$ such that $\sigma_{13,i}(\lambda_1(w)) = \lambda_4(v)$ or there is $w \in V_2$ such that $\sigma_{23,i}(\lambda_2(w)) = \lambda_4(v)$. Take $x_4 \in \mathcal{T}_A/\sim$ the equivalence class of s_4. We have $x_1 \preceq x_4$, $x_2 \preceq x_4$, $x_4 \preceq x_3$ and $\gamma(x_4) \leq \gamma(x_1) + \gamma(x_2)$. □

Intuitively, take the example of the library and consider a state s_1 such that $\gamma(s_1) = (1, 1)$ and where the member m_1 borrowed the book b_1 and a state s_2 such that $\gamma(s_2) = (1, 1)$ and where the member m_2 has returned the book b_2, then clearly s_1 and s_2 have upper bounds. We can construct a "little" state s_3 such that $\gamma(s_3) = (2, 2)$ including the two different cases.

Now, for each PASTD, we can determine a $c \in \mathbb{N}^n$ which is a bound satisfying Condition 4 of Definition 14, using function μ defined as follows.

Definition 19. *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD. The function μ, which gives for each PASTD expression F a vector of natural $c \in \mathbb{N}^n$, is defined recursively as follows.*

1. $\mu(\epsilon) = (0, ..., 0)$

2. $\mu(\mathcal{A}[q_1[F_1], ..., q_j[F_j]]) = sup(\{\mu(F_i) \mid 1 \leq i \leq j\})$

3. $\mu(||_\Delta[F_1, F_2]) = \mu(F_1) + \mu(F_2)$

4. $\mu(\,|_{x \in X}[F]) = \begin{cases} \mu(F) & \text{if } X \notin \{T_1, ..., T_n\} \\ \mu(F) + (u_1, ..., u_n) & \text{if } X = T_i \\ & \quad \text{where } u_i = 1 \text{ and} \\ & \quad u_j = 0 \text{ for all } i \neq j \end{cases}$

5. $\mu(\,|||_{x \in X}[F]) = \begin{cases} |X| \cdot \mu(F) & \text{if } X \notin \{T_1, ..., T_n\} \\ \mu(F) + (u_1, ..., u_n) & \text{if } X = T_i \\ & \quad \text{where } u_i = 1 \text{ and} \\ & \quad u_j = 0 \text{ for all } i \neq j \end{cases}$

Note that for all $C \subset \mathbb{N}^n$, we denote by $sup(C)$ the supremum of the set C in $\mathbb{N}^n$.

The function μ determines the maximum number of instances of each type that are involved in a transition. Intuitively, we count one instance for each quantified operator associated to the corresponding parameter. Remark that there will be at least one instance for each parameter that is used in a quantified operator. For a PASTD $A = (F, \overrightarrow{T}, \overrightarrow{P})$, $\mu(F)$ allows us to prove the backward-downward monotony.

Lemma 9. *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD and $c = \mu(F)$. For each transition $x_1 \to x_2$, there is $y_1 \to y_2$ such that $y_2 \preceq x_2$, $\gamma(y_2) \leq c$, and $\forall z_2 \in \mathcal{T}_A/\!\sim \cdot\, y_2 \preceq z_2 \preceq x_2 \implies \exists z_1 \in \mathcal{T}_A/\!\sim \cdot\, z_1 \to z_2 \wedge y_1 \preceq z_1 \preceq x_1$.*

Proof. By induction on the structure F of the PASTD. See the detailed proof in Appendix A. □

For instance, consider the PASTD of the library $(F, (T_m, T_b), (P_m, P_b))$ whose expression F is illustrated in Figure 1.2. By Definition 19, we have $\mu(F) = (2, 2)$ (remark that each parameter appears twice in the tree). It means that for each transition in the system, a maximum of 2 members and 2 books will be actually involved. The property is verified in the transition depicted in Figure 1.5: the lesser transition is represented by bold strokes and includes only one member and one book.

We can now state that PASTDs are effective RMTSs.

Theorem 5. *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD and $\tilde{\mathcal{S}}_A = (\mathcal{T}_A/\!\sim, \to, \preceq)$ the quotient MTS, then $(\mathcal{T}_A/\!\sim, \to, \preceq, \gamma, \mu(F), \overrightarrow{0})$ is an effective RMTS.*

Proof. Let $\mathcal{S} = (\mathcal{T}_A/\!\sim, \to, \preceq, \gamma, \mu(F), \overrightarrow{0})$. $(\mathcal{T}_A/\!\sim, \to, \preceq)$ is a MTS by Proposition 11 and $\preceq$ a po on $\mathcal{T}_A/\!\sim$.

1. Let $x, y \in \mathcal{T}_A/\!\sim$ such that $x \preceq y$. We clearly have $\gamma(x) \leq \gamma(y)$ by definition of γ. Thus, γ is monotone. Let $r \in \mathbb{N}^n$ be a specific rank. Remark that there exists a $k \in \mathbb{N}$ such that for all $x \in \gamma^{-1}(r)$ with $\tilde{g}(x) = (V, E, \lambda)$, $|V| \leq k$. Then, for all $x \in \gamma^{-1}(r)$ with $\tilde{g}(x) = (V, E, \lambda)$, $|V|$ and $|E|$ are bounded and $Im(\lambda)$ is always finite. By a combinatorial argument, there is a finite number of possible graphs whose rank is r. And because $\tilde{g}$ is a bijection, $\gamma^{-1}(r)$ is then finite.

2. Condition 2 is proved by Lemma 8.
3. For all $x \to y$, there exists $z \preceq x$ such that $z \to y$ and $\gamma(z) \leq \gamma(y) + \overrightarrow{0}$ because $\gamma(z) = \gamma(y)$ by Proposition 12.
4. Condition 4 is proved by Lemma 9.

As a consequence, $\mathcal{S}$ is an RMTS. The transition relation is decidable as it is defined by a set of rules in [23]. Also by definition, $\preceq$ is decidable and $\gamma(\tilde{s})$ computable for $s \in \mathcal{T}_A$. For $r \in \mathbb{N}^n$, $\gamma^{-1}(r)$ is computable by enumerating all well-formed states $\tilde{s}$ such that $\gamma(\tilde{s}) = r$. Thus, $\mathcal{S}$ is an effective RMTS. □

For any PASTD, we conclude that we can compute a finite pred-basis of any state s. However, if we want to compute a finite basis for $Pred^*(\uparrow s)$ and decide coverability, we need the wqo condition.

Theorem 6. *Bounded-PASTDs are WSTSs with effective pred-basis. Hence, the coverability problem is decidable.*

Proof. Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a Bounded-PASTD, $(\mathcal{T}_A/\sim, \to, \preceq, \mu(F), \overrightarrow{0})$ the corresponding RMTS, $\tilde{I} \subseteq \mathcal{T}_A/\sim$ and $\tilde{s} \in \mathcal{T}_A/\sim$. We have that $\preceq$ is a wpo as A is a Bounded-PASTD, thus we can compute a basis for $Pred^*(\uparrow\tilde{s})$. We can check whether $Pred^*(\uparrow\tilde{s}) \cap \tilde{I}$ is empty or not because $\tilde{I}$ is easily recognizable (initial states are explicited in [23]). Hence, by Proposition 10, $cov(\tilde{\mathcal{S}}_A, \tilde{I}, \tilde{s})$ is decidable. □

1.5 Related Work

Several models in the literature allow the specification of parameterized systems [35, 25, 40, 11, 33, 30], but most of them are too restrictive for some complex systems like information systems because they do not allow us to model relationships between entities or parameters. In general, process algebras [4, 7, 21] are suitable to specify information systems but it is not always clear how to handle parameterized systems. In particular, [22] shows that specifying and verifying information systems can be done for finite state systems. However, most of the existing models are not adapted to parameterized verification. The verification of EB^3 specifications, from which the ASTD notation derives, has been studied in [41] but the paper does not consider parameters. In [40], the authors present a model close to ours called *LTS schema* that improves labeled transition systems with process algebra operators and allows parameterization.

Along with abstraction [32, 10, 2] and cut-off techniques [16, 25, 40, 39, 29, 2], Well Structured Transition Systems has been widely used to verify parameterized systems [33, 30, 11, 2] or more generally infinite systems. Finding the right wqo on the set of states is a central issue. In [33], [11] and [30], the authors use a graph representation of their states and the adequate subgraph wqo that is similar to our case: the set of states is restricted to those of bounded simple paths. Alternate wqo on graphs, like the *graph minor* relation [36], can also be used [30]. But in our case it does not satisfies the monotony condition.

In [30], the authors present a framework for viewing *Graph Transformation Systems* as WSTSs under certain conditions. They give as well an algorithm computing pred-basis in this particular context. The framework of *Nicely Sliceable WSTS* [6], which is closer to ours in term of requirements, gives also a sub-class of WSTS. However, they are both tied to a wqo condition (by definition of WSTS), unlike the RMTS framework.

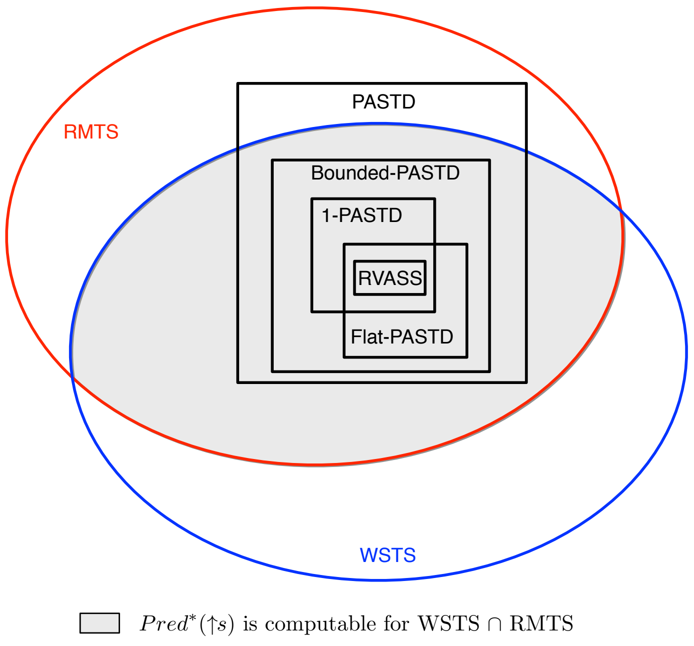


Figure 1.14: Comparison of the models

1.6 Conclusion

Results and discussion. In this article, we have presented a model called PASTD, a variant of ASTD, which allows for the specification of complex parameterized systems like information systems. A PASTD specification is a formal model with a graphical notation that is easy to read. PASTDs are more expressive than VASSs with resets, hence the reachability problem is undecidable for PASTDs. Furthermore, we pointed out some subclasses of PASTD (Bounded-PASTD, 1-PASTD and Flat-PASTD) that are WSTSs. Then, we introduced a new general framework called RMTS to prove the existence and computability of pred-basis in any monotone transition system. RMTSs give a set of properties that should be satisfied by a system to compute a pred-basis but do not require a wqo as in WSTSs. Finally, we showed that PASTDs are RMTSs and thus that the coverability problem is decidable for Bounded-PASTDs. The relation between the different classes of model presented in this paper is summarized in Figure 1.14. Remark that to match with the definition of RMTS, we assume that all quasi-orderings are partial orderings (by considering quotient space for instance).

For Bounded-PASTDs, we have a complete procedure to decide coverability and thus to verify safety properties. We think that Bounded-PASTDs can model a larger class of parameterized systems than classical models like RVASSs. Unfortunately, many parameterized systems are not Bounded-PASTDs. Indeed, the library example presented in this paper is not well-structured as we can find an example of antichain similar to the one of Figure 1.9. However, we were able to prove that PASTDs are effective RMTSs.

In fact, the RMTS framework allowed us to find the adequate conditions to construct the PASTD class of systems that is RMTS. For instance, considering abstract states in PASTDs was necessary to prove the backward-downward monotony, thus to prove the effective pred-basis without wqo and can make the computation easier in the same time. Intuitively, an abstract state is a state where the local configurations of some instances are unknown. It is useful to model a quantified interleaving state with partial information. Hence, an abstract state represents a set of concrete states whose quantified interleaving are completely instantiated. This allows for the factorization of the pred-basis computation (see Appendix A for more details).

Considering an effective RMTS $\mathcal{S} = (Q, \rightarrow, \leq, \gamma, c, d)$ and a state $s \in Q$, a semi-decision procedure to determine if an initial state belongs to $Pred^*(\uparrow s)$ is possible. More than a semi-algorithm, sometimes the procedure may terminate without wqo even if s is not coverable. This is the case when a basis for $Pred^*(\uparrow s)$ is computed. Indeed, even if $(Q, \leq)$ is not a wqo, $(Pred^*(\uparrow s), \leq)$, which depends on the state s, may be wqo. That justifies the practical importance of this semi-decidability result. Moreover, determining the complexity of a wqo-based algorithm is not easy [38]. In practice, if we are not able to get a reasonable complexity, the wqo argument may not be sufficient to guarantee that the implemented verification procedure will end.

Future work. Experimenting the backward algorithm on different RMTSs like PASTDs would be the next step in order to determine for which kind of systems and coverability properties the procedure is likely to stop within a reasonable amount of time. Besides, concerning PASTDs, we have given two easily recognizable subclasses of Bounded-PASTDs but we should be able to determine a more general algorithm deciding if a PASTD is well-structured, which is hinted by Lemma 5. Finally, as computing $Pred^*(\uparrow s)$ should not be necessary to solve the coverability problem, it would be interesting to find other termination conditions than wqo for a backward algorithm that checks emptiness of the intersection of $Pred^*(\uparrow s)$ and the set of initial states; some conditions that would be more dependent on the transition system.

Appendix A

Parameterized ASTD

A.1 Preliminaries

A *Transition System* (TS) is a pair $S = (Q, \rightarrow)$, where Q is a set (of states) and $\rightarrow \subseteq Q \times Q$ is the set of transitions between states. We write $q \rightarrow q'$ for $(q, q') \in \rightarrow$, and $q \xrightarrow{*} q'$ if either $q = q'$ or there exists a finite sequence of transitions $q \rightarrow q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q'$, called a path. We define $Pred(q) = \{q' \in Q \mid q' \rightarrow q\}$ as the set of immediate predecessors of q and $Pred^*(q) = \{q' \in Q \mid q' \xrightarrow{*} q\}$ as the set of all predecessors of q.

A *quasi-ordering* (qo) is a reflexive and transitive binary relation $\leq$ on a set X; we also say that $(X, \leq)$ is a qo. A *partial ordering* (po) is an antisymmetric qo.

Let $(X, \leq)$ be a po. For all $s_1, s_2 \in X$, we say that $s_3 \in X$ is the *supremum* (or sup) of s_1 and s_2 noted $sup(s_1, s_2)$ if s_3 is an upper bound of s_1 and s_2 (*i.e.* $s_1 \leq s_3$ and $s_2 \leq s_3$), and for all upper bounds $s_4 \in X$, we have $s_3 \leq s_4$.

Let $\leq$ be a qo on a set X. An *upward-closed* set is a subset $Y \subseteq X$ such that if $x \leq y$ and $x \in Y$ then $y \in Y$. For some $x \in X$, we write $\uparrow x = \{y \in X \mid x \leq y\}$ its upward-closure, and for some $Y \subseteq X$, $\uparrow Y = \bigcup_{x \in Y} \uparrow x$. A *basis* of an upward-closed subset $Y \subseteq X$ is any set B such that $Y = \uparrow B$. We say that an upward closed set Y has a *finite basis* if there exists some finite basis B of Y.

A qo $(X, \leq)$ is *well-founded* if there is no infinite sequence $(x_n)_{n \in \mathbb{N}}$ over X such that $x_{i+1} \leq x_i$ for every $i \in \mathbb{N}$. It is a *well-quasi-ordering* (wqo) if every infinite sequence $(x_n)_{n \in \mathbb{N}}$ over X contains an increasing pair, i.e. $\exists i < j$ such that $x_i \leq x_j$. If the wqo is a po, then it is called a *well partial order* (wpo). An *antichain* is a set in which each pair of different elements is incomparable.

We call *permutation* any bijection $f : X \rightarrow X$. We denote by id_X the identity on the set X. We denote by $dom(f)$ the domain of the function $f : X \rightarrow Y$, $ran(f)$ its image , *i.e.* $ran(f) = \{y \in Y \mid \exists x \in X \cdot f(x) = y\}$, $X \lhd f$ the domain restriction of the function to X and by $f \mathbin{\lhd\!\!\!+} g$ the overriding of $f : X \rightarrow Y$ by $g : W \rightarrow Y$, *i.e.* $(f \mathbin{\lhd\!\!\!+} g)(x) = g(x)$ if $x \in W$ and $(f \mathbin{\lhd\!\!\!+} g)(x) = f(x)$ otherwise.

A.2 A fragment of the ASTD language

Algebraic State-Transition Diagram (ASTD) [24] is a graphical notation combining automata, statecharts and process algebra to describe complex dynamic systems like information systems. They are closely related to process algebras like CSP [27], CCS [34], ACP [4], LOTOS [7] and EB^3 [21].

Essentially, ASTD are like a process algebra with hierarchical automata as elementary process expressions. Automata can be combined freely with process algebra operators. ASTD have a structured operational semantics in the Plotkin style, which was first used by Milner for CCS and later on for LOTOS and CSP [37]. They are recursively defined structures and include many types like automaton, synchronization, quantified choice and quantified interleaving.

ASTD are useful to model information systems as they provide a concise, visual and formal mechanism for specifying all the scenarios of an information system [24]. For example, they make explicit the handling of entity instances by using quantifications. Furthermore, the model has been used in [15] to specify access control and security policies.

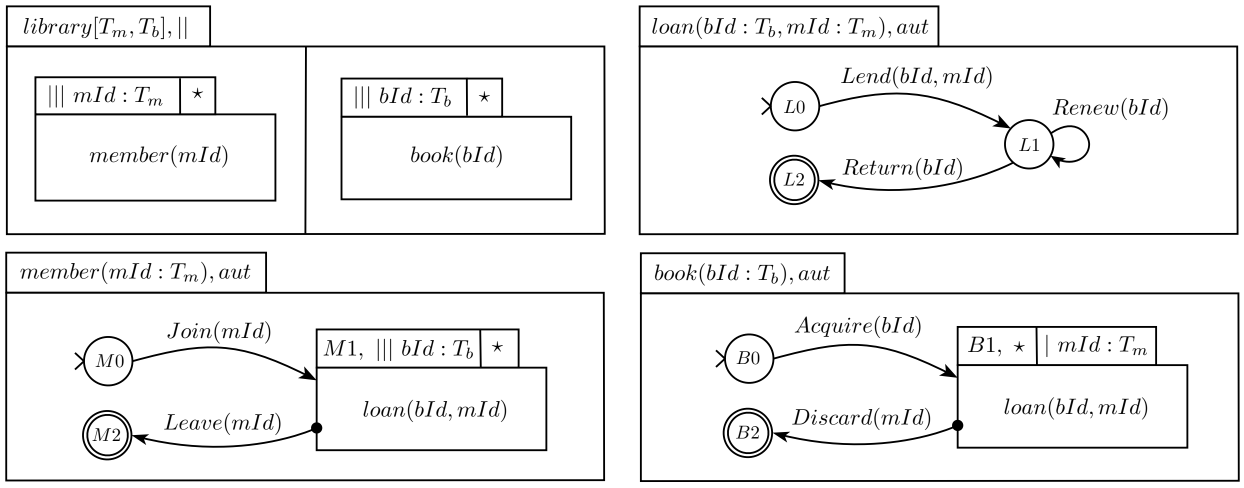


Figure A.1: Example of a library system

For instance, an ASTD model of a library system is given in Figure A.1. The system manage loans of books by members. The ASTD on the top left hand corner, whose name is *library*, is a synchronization between two other ASTD which consist in a quantified interleaving on the set T_m of members for the the process *member* and a quantified interleaving on the set T_b of books for the process *book*. The process $book(bId)$ is described by the ASTD on the bottom right corner. A book is acquired by the library. It can be discarded if it is not lent. If acquired, a book bId can "choose" a member m to run the process $loan(bId, mId)$. The member process is similar except that a member m runs the $loan(bId, mId)$ process for every book.

Definition 20. *ASTD are defined inductively and includes the following types:* Automaton*,* Kleene Closure*,* Choice*,* Synchronization*,* Quantified Choice*,* Quantified Interleaving*,* ASTD Call *and* Elementary ASTD*. The signature of an ASTD is given by a tuple whose first element is a tag corresponding to the ASTD type.*

1. *The Elementary ASTD* $\langle \mathsf{elem} \rangle$ *is the simplest ASTD that can be defined. It is used to represent some state of the Automaton ASTD.*

2. *An Automaton ASTD is similar to a traditional automaton, except that its states can be of any ASTD type. It is denoted by* $\langle \mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0 \rangle$*, where name is the name of the ASTD structure,* Σ *a set of events,* N *a set of state names,* ν *a function that maps each state name to an ASTD,* δ *a transition relation,* $SF \subseteq N$ *a set of shallow final states,*

$DF \subseteq N$ a set of deep final states and $n_0 \in N$ an initial state. The transition relation δ is given by a set of tuples $\langle n_1, n_2, \sigma, final? \rangle$, where n_1, n_2 are two state names, $\sigma \in \Sigma$ is an event and $final?$ is a boolean denoting a transition that can be fired only from a final state (the definition of a final ASTD state is given in the sequel). An event is noted $l(v_1, \ldots, v_n)$ where l is called the event label, and v_i are event parameters. Function α extracts the label of an event: $\alpha(l(v_1, \ldots, v_n)) = l$. By extension, $\alpha(a)$ returns the set of labels occurring in ASTD a.

3. *The Kleene Closure is given by a tuple $\langle \star, n, b \rangle$, where n is the ASTD name and b an ASTD. It allows for iteration on ASTD b a finite number of time (including zero). An iteration is completed when the component ASTD has reached a final state.*

4. *A Choice ASTD $\langle |, n, l, r \rangle$, where n is the ASTD name, allows a choice between two component ASTD l and r like in a process algebra.*

5. *A Synchronization ASTD $\langle ||, n, \Delta, l, r \rangle$ between two component ASTD l and r describes a behavior where l and r run concurrently by executing events, whose labels are in Δ, at the same time and interleaving the other events. We denote by $||$ the Synchronization ASTD where $\Delta = \alpha(l) \cap \alpha(r)$.*

6. *A Quantified Choice ASTD $\langle |:, n, x, T, b \rangle$ allows to pick a value v from a finite set T and execute component ASTD a where every occurrence of the variable x is replaced by the value v.*

7. *A Quantified Interleaving ASTD $\langle |||:, n, x, T, b \rangle$ allows for executing as many interleaving instances of ASTD b as the number of values in a finite set T. Each instance is executed such that x is replaced by the corresponding value.*

8. *Finally, it is possible to call an ASTD defined in another diagram together with some parameters, but not recursively in our case. According to this restriction, we consider ASTD Calls as a syntactic sugar, that is we can always replace a call by the called ASTD. In the following, we do not consider ASTD Calls for the sake of simplicity.*

Besides, in order to represent a concrete system, an ASTD must not contain any free variable. In such ASTD, we consider that every variable used in a parameterized event is bound either by a Quantified Choice or a Quantified Interleaving.

Figure A.2 shows some examples of ASTD in their graphical notations. For example, the ASTD on the top left hand corner is an Automaton ASTD, whose name is *processA*. It is composed of two states and one transition labeled by the event b. The first state $S1$ is itself an Automaton ASTD and the second one $S2$ is an Elementary ASTD. The symbol $>$ denotes the initial automaton state and the double circle a final automaton state. The transition that is decorated by a bullet at its source means that its boolean $final?$ is true, *i.e.* it can be fired only if $S1$ is in its final state $Q2$. An example of an ASTD with free variable is the ASTD *processB*(v) that contains the variable v. As $c(v)$ is a parameterized event, we must instantiate *processB*(v) with a value to obtain a concrete event. The ASTD *processC* is a Quantified Interleaving ASTD that calls the *processB*(v) ASTD and binds the variable v to its quantified operator. The ASTD *main*, that synchronizes two sub-ASTD, is another example of an ASTD Call. Note that naming the sub-ASTD is not always necessary and that we can coalesce ASTD boxes when the outer ASTD is a unary operator.

The coalescing is indicated by adding the tab of the inner ASTD to the outer unary one, like in $processB(v)$.

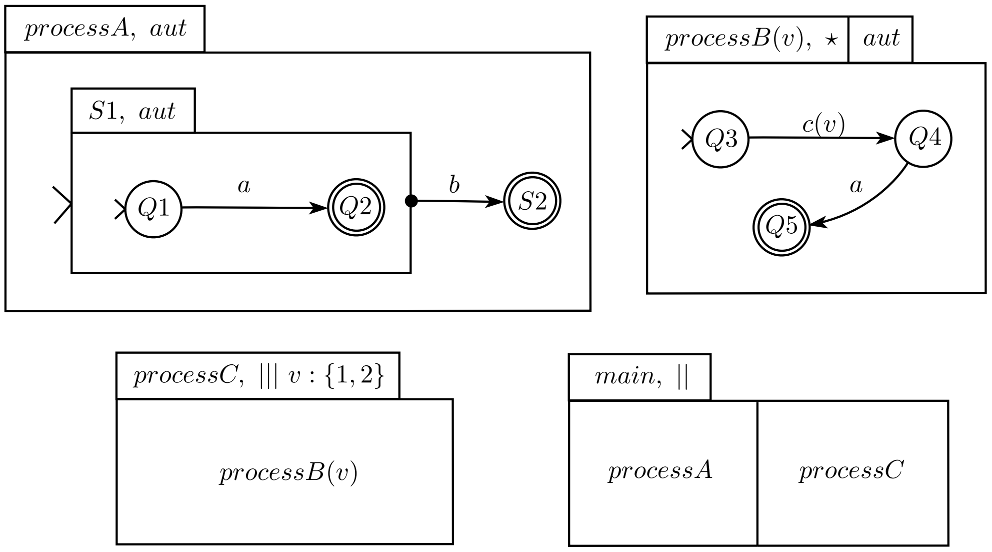


Figure A.2: Example of ASTD

The main differences with the ASTD described in [23] are that Automata are simplified, that there is no Sequence ASTD, no Guard ASTD and that Quantified Synchronizations are replaced by Quantified Interleavings. All definitions in this section are slightly modified from [23] according to the previous considerations on the ASTD language.

As an ASTD describes a dynamical system, for each ASTD, we can define a set of ASTD states. ASTD states are given by the following definition.

Definition 21. *A state expression of an ASTD is given inductively by an expression according to the following types, where the first component is a label identifying the state type (since an ASTD state is a sum type):*

1. $\langle \mathsf{elem}_\circ \rangle$*, the* elementary state*, which is the only type of state for an Elementary ASTD;*
2. $\langle \mathsf{aut}_\circ, n, s \rangle$*, an* automaton state*, where n is the name of a state of the automaton and s a sub-state;*
3. $\langle \star_\circ, started?, [\bot \mid s] \rangle$*, a* Kleene closure state*, with a boolean $started?$ indicating whether the first iteration has been started, and where s is a sub-state, and $\bot$ is used instead of s when $started?$ is* false*;*
4. $\langle |_\circ, [\bot \mid \mathsf{left} \mid \mathsf{right}], [\bot \mid s] \rangle$*, a* choice state*, with a variable indicating which choice has been made if it has been and where s is a sub-state;*
5. $\langle ||_\circ, s_l, s_r \rangle$*, a* synchronization state*, where s_l and s_r are sub-states;*
6. $\langle |:_\circ, [\bot \mid v], [\bot \mid s] \rangle$*, a* quantified choice state*, where v is the value of the quantified choice (if it has been made) and s a sub-state;*

7. $\langle |||{:}_\circ, f\rangle$, *a* quantified interleaving state*, where f is a function, whose domain of definition is a non-empty and finite set of values and whose codomain is the set of ASTD states.*

For example, a state of the ASTD *processC* from Figure A.2 can be given by the expression $(|||{:}_\circ, \{1 \mapsto (\star_\circ, \text{true}, (\mathsf{aut}_\circ, Q4, (\mathsf{elem}_\circ))), 2 \mapsto (\star_\circ, \text{false}, \bot)\})$, which means that $processB(1)$ is in state $Q4$ and $processB(2)$ has not yet started.

Some states of an ASTD may be initial or final. This is specified by a function *init* that returns the initial state of an ASTD and a predicate *final* that determines whether a state is considered as final.

Definition 22. *Let a be an ASTD. The function init, which returns the initial state of a, is defined as follows:*

1. *$init((\mathsf{elem})) = (\mathsf{elem}_\circ)$, the elementary state;*
2. *$init((\mathsf{aut}, n, \Sigma, N, \nu, \delta, SF, DF, n_0)) = (\mathsf{aut}_\circ, n_0, init(\nu(n_0)))$, the automaton state, whose name n_0 is the initial automaton state's one, and whose sub-state is define recursively by the initial state of n_0;*
3. *$init((\star, n, b)) = (\star_\circ, \text{false}, \bot)$, the Kleene closure state, where the first iteration has not been started;*
4. *$init((|, n, l, r)) = (|_\circ, \bot, \bot)$, the choice state, where the choice has not been made;*
5. *$init((||, n, \Delta, l, r)) = (||_\circ, init(l), init(r))$, the synchronization state, where the two component states are in their initial states;*
6. *$init((|{:}, n, x, T, b)) = (|{:}_\circ, \bot, \bot)$, the quantified choice state, where the choice has not been made;*
7. *$init((|||{:}, n, x, T, b)) = (|||{:}_\circ, T \times \{init(b)\})$, the quantified interleaving state, where each component state is set to its initial state.*

Definition 23. *Let s be an ASTD state and a an ASTD of the same type. The predicate final(s), which determines if s is final in a, is defined as follows:*

1. *If $s = (\mathsf{elem}_\circ)$, then $final(s) \equiv \text{true}$, the elementary state is final in the Elementary ASTD;*
2. *If $s = (\mathsf{aut}_\circ, n, ss)$ and $a = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)$, then $final(s) \equiv (n \in DF \wedge final(ss)) \vee n \in SF$,* i.e. *the automaton state s is final in the Automaton ASTD a either if the state is marked as deep final and the corresponding sub-state ss is final or if the state is marked as shallow final;*
3. *If $s = (\star_\circ, started?, ss)$, then $final(s) \equiv final(ss) \vee \neg started?$,* i.e. *s is final either if the sub-state ss is final or if the first iteration has not been started;*
4. *If $s = (|_\circ, \bot, \bot)$ and $a = (|, n, l, r)$, then $final(s) \equiv final(init(l)) \vee final(init(r))$,* i.e. *s is final if either the initial state of the left component ASTD or the initial state of the right component is final;*
5. *If $s = (|_\circ, \_, ss)$, then $final(s) \equiv final(ss)$,* i.e. *s is final if the sub-state ss is final;*

6. *If $s = (||_\circ, s_l, s_r)$, then final$(s) \equiv$ final$(s_l) \wedge$ final(s_r),* i.e. *s is final if both sub-states are final;*

7. *If $s = (|:_\circ, \bot, \bot)$, then final$(s) \equiv$ final$(init(b))$,* i.e. *s is final if the initial state of the component ASTD is final;*

8. *If $s = (|:_\circ, v, ss)$ with $v \neq \bot$, then final$(s) \equiv$ final(ss),* i.e. *s is final if the sub-state ss is final;*

9. *If $s = (|||:_\circ, f)$ and $a = (|||:, n, x, T, b)$, then final$(s) \equiv \forall v \in T \cdot$ final$(f(v))$,* i.e. *s is final if each sub-state is final.*

Consider the example of the ASTD *processA* of Figure A.2. The initial state of the ASTD is $(\mathsf{aut}_\circ, S1, (\mathsf{aut}_\circ, Q1, (\mathsf{elem}_\circ)))$ and the only final state is $(\mathsf{aut}_\circ, S2, (\mathsf{elem}_\circ))$.

As seen previously, Automaton ASTD can be defined with free variables, which are used in parameterized events. Those variables can be bound by some quantified operator ASTD. In order to keep track of the bound variables in a sub-state, when computing a transition, we need an execution environment that contains a valuation for each variable.

Definition 24. *An environment is a function which maps a variable to a value. An environment is noted $(\![x_1, \ldots, x_n := v_1, \ldots, v_n]\!)$, or $(\![\vec{x} := \vec{v}]\!)$. An empty environment is noted $(\![]\!)$. An environment Γ can be used in a substitution. The symbol $\lhd$ is a composition operator on environments such that if Γ_1, Γ_2 are environments and u an expression, $u[\Gamma_1 \lhd \Gamma_2] = (u[\Gamma_1])[\Gamma_2]$. Note that Γ_1 has precedence over Γ_2 when $\Gamma_1 \lhd \Gamma_2$ is used in a substitution.*

For example, if we consider the ASTD *processC* from Figure A.2, and when looking locally at the sub-state $(\star_\circ, \text{true}, (\mathsf{aut}_\circ, Q3, (\mathsf{elem}_\circ)))$ in *processB*(1), we need the environment $(\![v := 1]\!)$ to compute the transition $c(1)$, because the valuation does not appear in the expression of the sub-state.

Remark that the notion of environment does not affect the definitions of the function *init* and the predicate *final*. However, it is important in the definition of the transition relation of ASTD. The operational semantics of ASTD consists of a set of inference rules defined inductively. We write transitions with respect to an environment Γ, noted as $s \xrightarrow{\sigma,\Gamma}_a s'$, where s and s' are two states of an ASTD a and σ an event. Subscript a can be omitted when it is clear from the context which ASTD is being referred to.

Definition 25. *Let a be an ASTD without free variables and s and s′ two states of a. We compute a transition starting from an empty environment, using the following inference rule.*

$$\mathsf{env}\ \frac{s \xrightarrow{\sigma,(\![]\!)} s'}{s \xrightarrow{\sigma} s'}$$

The rule env *allows to introduce the environment in transitions, that is necessary to use the other inference rules described below. Let a be an ASTD, Γ an environment and σ an event. The operational semantics is inductively defined for each ASTD subtype.*

1. *If $a = (\mathsf{aut}, n, \Sigma, N, \nu, \delta, SF, DF, n_0)$, each transition in a is given by one of the two following inference rules:*

$$\mathsf{aut}_1\ \frac{\delta(n_1, n_2, \sigma', final?) \qquad final? \Rightarrow final(s) \wedge \sigma'[\Gamma] = \sigma}{(\mathsf{aut}_\circ, n_1, s) \xrightarrow{\sigma,\Gamma} (\mathsf{aut}_\circ, n_2, init(\nu(n_2)))}$$

$$\mathsf{aut}_2 \frac{s \xrightarrow{\sigma,\Gamma}_{\nu(n)} s'}{(\mathsf{aut}_\circ, n, s) \xrightarrow{\sigma,\Gamma} (\mathsf{aut}_\circ, n, s')}$$

Rule aut_1 *describes a transition between local states. There is a transition, labeled by* σ *and with environment* Γ*, between some state of* n_1 *and the initial state of* n_2 *if there is an arrow in the automaton between* n_1 *and* n_2*, labeled by* σ' *and such that the environment applied as a substitution on* σ' *gives* σ*, and if the transition marked as final, then the source state must be a final state. Rule* aut_2 *handles transition within a sub-state.*

2. *If* $a = (\star, n, b)$*, the inference rules are:*

$$\star_1 \frac{(\mathit{final}(s) \vee \neg \mathit{started?}) \qquad \mathit{init}(b) \xrightarrow{\sigma,\Gamma}_b s'}{(\star_\circ, \mathit{started?}, s) \xrightarrow{\sigma,\Gamma} (\star_\circ, \mathrm{true}, s')}$$

$$\star_2 \frac{s \xrightarrow{\sigma,\Gamma}_b s'}{(\star_\circ, \mathrm{true}, s) \xrightarrow{\sigma,\Gamma} (\star_\circ, \mathrm{true}, s')}$$

Rule $\star_1$ *allows for (re-)starting from the initial state of the component ASTD when a final state has been reached or for the first iteration. Rule* $\star_2$ *allows for execution on the component ASTD when an iteration has been started.*

3. *If* $a = (|, n, l, r)$*,*

$$|_1 \frac{\mathit{init}(l) \xrightarrow{\sigma,\Gamma}_l s'}{(|_\circ, \bot, \bot) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathsf{left}, s')} \qquad |_2 \frac{\mathit{init}(r) \xrightarrow{\sigma,\Gamma}_r s'}{(|_\circ, \bot, \bot) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathsf{right}, s')}$$

$$|_3 \frac{s \xrightarrow{\sigma,\Gamma}_l s'}{(|_\circ, \mathsf{left}, s) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathsf{left}, s')} \qquad |_4 \frac{s \xrightarrow{\sigma,\Gamma}_r s'}{(|_\circ, \mathsf{right}, s) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathsf{right}, s')}$$

Rules $|_1$ *and* $|_2$ *deal with the execution of the first event from the initial state. Rules* $|_3$ *and* $|_4$ *deal with the execution of the subsequent events from the chosen component.*

4. *If* $a = (||, n, \Delta, l, r)$*,*

$$||_1 \frac{\alpha(\sigma) \notin \Delta \qquad s_l \xrightarrow{\sigma,\Gamma}_l s_l'}{(||_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (||_\circ, s_l', s_r)} \qquad ||_2 \frac{\alpha(\sigma) \notin \Delta \qquad s_r \xrightarrow{\sigma,\Gamma}_r s_r'}{(||_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (||_\circ, s_l, s_r')}$$

$$||_3 \frac{\alpha(\sigma) \in \Delta \qquad s_l \xrightarrow{\sigma,\Gamma}_l s_l' \qquad s_r \xrightarrow{\sigma,\Gamma}_r s_r'}{(||_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (||_\circ, s_l', s_r')}$$

Rules $||_1$ and $||_2$ respectively describe execution of events with no synchronization required on the left-hand side and the right hand side of the synchronization ASTD. Rule $||_3$ describes the synchronization between the left hand side and the right hand side.

5. *If $a = (|:, n, x, T, b)$,*

$$|:_1 \frac{init(b) \xrightarrow{\sigma, ([x:=v]) \lhd \Gamma}_b s' \qquad v \in T}{(|:_\circ, \bot, \bot) \xrightarrow{\sigma,\Gamma} (|:_\circ, v, s')} \qquad |:_2 \frac{s \xrightarrow{\sigma, ([x:=v]) \lhd \Gamma}_b s' \qquad v \neq \bot}{(|:_\circ, v, s) \xrightarrow{\sigma,\Gamma} (|:_\circ, v, s')}$$

Rule $|:_1$ describes the execution of the first event from the initial state, whereas Rule $|:_2$ deal with the execution of the subsequent events. In both cases, the value bound to the quantification variable is added to the execution environment and can be used to make the proof after the environment has been applied as a substitution.

6. *If $a = (|||:, n, x, T, b)$,*

$$|||:_1 \frac{f(v) \xrightarrow{\sigma, ([x:=v]) \lhd \Gamma}_b s'}{(|||:_\circ, f) \xrightarrow{\sigma,\Gamma} (|||:_\circ, f \lhd\!\!\!- \{v \mapsto s'\})}$$

Rule $|||:_1$ describes the execution of an event from the component ASTD instantiated by value v. The value v bound to the quantification variable x is added to the execution environment.

For example, consider the ASTD *processC* from Figure A.2 and let $s = (|||:_\circ, \{1 \mapsto (\star_\circ, \text{true}, (\mathsf{aut}_\circ, Q4, (\mathsf{elem}_\circ))), (\star_\circ, \text{false}, \bot)\})$ be a state of *processC*. Let us prove that there is a transition labeled by $c(1)$ between the initial state of *processC* and s. We have $init(processC) = (|||:_\circ, \{1 \mapsto (\star_\circ, \text{false}, \bot), 2 \mapsto (\star_\circ, \text{false}, \bot)\})$. We can derive the transition as follows:

$$\mathsf{env}\frac{|||:_1 \dfrac{\star_1 \dfrac{\neg started? \qquad \mathsf{aut}_1 \dfrac{\delta(Q3, Q4, c(v), \text{true}) \qquad \mathit{final}((\mathsf{elem}_\circ)) \land c(v)[([v := 1])] = c(1)}{(\mathsf{aut}_\circ, Q3, (\mathsf{elem}_\circ)) \xrightarrow{c(1), ([v:=1])} (\mathsf{aut}_\circ, Q4, (\mathsf{elem}_\circ))}}{(\star_\circ, \text{false}, \bot) \xrightarrow{c(1), ([v:=1])} (\star_\circ, \text{true}, (\mathsf{aut}_\circ, Q4, (\mathsf{elem}_\circ)))}}{init(processC) \xrightarrow{c(1), ([])} s}}{init(processC) \xrightarrow{c(1)} s}$$

To make the state expressions easier to read, we introduce a graphical representation for the ASTD states. An ASTD state can be represented by a tree where each node is labeled by a tuple giving the type of the state. If a type of state includes a sub-state in its definition, then it is represented by a child node. Furthermore, we split up the nodes for quantifications so that the values appear in child nodes. A labeled tree is given by an expression thanks to the grammar $Tree ::= Node \mid Node\langle Tree, ..., Tree\rangle$.

Definition 26. *The tree representation of an ASTD state is given by the tree function, which is defined inductively as follows:*

1. *$tree((\mathsf{aut}_\circ, n, (\mathsf{elem}_\circ))) = (\mathsf{aut}_\circ, n)$*

2. $tree((\mathsf{aut}_\circ, n, s)) = (\mathsf{aut}_\circ, n)\langle tree(s)\rangle \qquad$ *if* $s \neq (\mathsf{elem}_\circ)$

3. $tree((\star_\circ, started?, s)) = \begin{cases} \star_\circ & \textit{if } started? = \text{false} \\ \star_\circ\langle tree(s)\rangle & \textit{else} \end{cases}$

4. $tree((|_\circ, side, x)) = \begin{cases} |_\circ & \textit{if } side = \bot \\ (|_\circ, side)\langle tree(x)\rangle & \textit{else} \end{cases}$

5. $tree((||_\circ, s_l, s_r)) = ||_\circ\langle tree(s_l), tree(s_r)\rangle$

6. $tree((|:_\circ, v, x)) = \begin{cases} |:_\circ & \textit{if } v = \bot \\ |:_\circ\langle v\langle tree(x)\rangle\rangle & \textit{else} \end{cases}$

7. $tree((|||:_\circ, f)) = |||:_\circ\langle v_1\langle tree(f(v_1))\rangle, ..., v_k\langle tree(f(v_k))\rangle\rangle$ *where* $\bigcup_i \{v_i\} = dom(f)$

The set of labels consists of a finite set of tuples for each type of ASTD state and many (possibly infinite) sets of values (used in the case of quantified operators).

For instance, the state

$$s = (|||:_\circ, \{1 \mapsto (\star_\circ, \text{true}, (\mathsf{aut}_\circ, Q4, (\mathsf{elem}_\circ))), 2 \mapsto (\star_\circ, \text{false}, \bot)\})$$

of *processC* from Figure A.2 is represented by a tree depicted in Figure A.3.

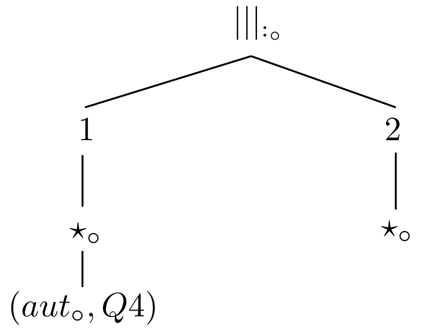


Figure A.3: Example of tree representation of a state

Remark that the tree branches are ordered in that representation, even for the quantified interleaving operator. We simply assume that the ordering of the branches of $dom(f)$ is arbitrary. Except from that, the function *tree* is a simple rewriting and it is obvious that for all ASTD state s, the tree representation $tree(s)$ is semantically equivalent to s. In the following, we may refer indifferently to s or $tree(s)$ as the state expression s.

A.3 Parameterized ASTD

Some of the most interesting features of the ASTD are the quantified operators (interleaving and choice) that allow to model replication of processes and complex interactions between them. Consider a simple library system handling members, books and loans. If we want to specify an ASTD for this system, we can use the Quantified Interleaving to model many replicated processes running concurrently, each process representing a specific member for example. In the original definition of quantified operator ASTD, we have to specify a constant set of quantification. This means that the number of members in the library, for example, is fixed. To model a library that works with any number of members, we need to consider a more abstract specification that takes the set of members as a parameter. The system can intuitively be parameterized by a number of members and of books. Therefore, we allow quantification sets to be variables in Quantified Choice et Quantified Interleaving. We call those new variables the parameters of the system. To this end, we introduce the new definition of Parameterized ASTD.

Definition 27 (PASTD)**.** *A* Parameterized ASTD *(PASTD) $a[T_1, ..., T_k]$ is a model of an ASTD equipped with some parameters $T_1, ..., T_k$. Each parameter T_i has a type P_i, called a* parameter domain*. A parameter can only be instantiated with a finite subset of its parameter domain. Parameter domains are disjoint. Moreover, no subset of a parameter domain can be used as a constant in the specification. We denote by $a[T_1 := V_1, ..., T_k := V_k]$ the ASTD corresponding to a PASTD instantiated by values $V_1 \subset P_1, ..., V_k \subset P_k$. Parameters are transferred to sub-PASTD and are used as sets of quantification for quantified choices or quantified interleavings.*

For instance, a PASTD model of a library system is given in Figure A.1. The system manage loans of books by members. There are two parameters T_m and T_b, with domains $P_m = \{m_1, m_2, ...\}$ and $P_b = \{b_1, b_2, ...\}$, respectively, representing the sets of member and book identifiers. A book is acquired by the library. It can be discarded if it is not lent. A member must join the library in order to borrow a book. She can leave the library membership when all her loans are returned.

In some parameterized systems, like Petri Nets, parameters are naturals numbers and denote the number of replicated processes. However, this abstraction cannot be made in our case. This is due to the interactions between the various instances in some complex parameterized systems. Indeed, consider the previous example of the library system. With natural numbers as parameters, we could specify the number of registered members or acquired books in the library but not the fact that a specific member m_1 has borrowed the book b_2. That is why, in PASTD, we use a finite set of identifiers instead of a natural number to instantiate a parameter. This said, we will see later how to take into account the symmetries induced by this modeling.

As we introduce free variables in the specification of PASTD, we need a valuation of those variables for the model to represent a concrete system. Consider an "additional environment" consisting of a vector of parameters values. Besides, parameters of PASTD are noted between brackets "[...]" and parameters of events and calls between parenthesis "(...)".

Definition 28 (IPASTD)**.** *If $a[\overrightarrow{T}]$ is a PASTD, we call $a[\overrightarrow{T} := \overrightarrow{V}]$ an* Instantiated PASTD (IPASTD)*. PASTD with parameters $\overrightarrow{T} = (T_1, .., T_k)$ are instantiated with values $\overrightarrow{V} = (V_1, ..., V_k)$ as follow:*

1. $(\mathsf{elem})[\overrightarrow{T} := \overrightarrow{V}] = (\mathsf{elem})$
2. $(\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T} := \overrightarrow{V}] =$
   $(\mathsf{aut}, name, \Sigma, N, \nu', \delta, SF, DF, n_0)$, *where* $\nu'(n) = \nu(n)[\overrightarrow{T} := \overrightarrow{V}]$
3. $(\star, n, b)[\overrightarrow{T} := \overrightarrow{V}] = (\star, n, b[\overrightarrow{T} := \overrightarrow{V}])$
4. $(|, n, l, r)[\overrightarrow{T} := \overrightarrow{V}] = (|, n, l[\overrightarrow{T} := \overrightarrow{V}], r[\overrightarrow{T} := \overrightarrow{V}])$
5. $(||, n, \Delta, l, r)[\overrightarrow{T} := \overrightarrow{V}] = (||, n, \Delta, l[\overrightarrow{T} := \overrightarrow{V}], r[\overrightarrow{T} := \overrightarrow{V}])$

6. $(|:, n, x, T_i, b)[\overrightarrow{T} := \overrightarrow{V}] = (|:, n, x, V_i, b[\overrightarrow{T} := \overrightarrow{V}])$, *where* $T_i \in \overrightarrow{T}$ *is a parameter and* $V_i \in \overrightarrow{V}$ *a value*

7. $(|:, n, x, W, b)[\overrightarrow{T} := \overrightarrow{V}] = (|:, n, x, W, b[\overrightarrow{T} := \overrightarrow{V}])$, *if* W *is not a parameter*

8. $(|||:, n, x, T_i, b)[\overrightarrow{T} := \overrightarrow{V}] = (|||:, n, x, V_i, b[\overrightarrow{T} := \overrightarrow{V}])$, *where* $T_i \in \overrightarrow{T}$ *is a parameter and* $V_i \in \overrightarrow{V}$ *a value*

9. $(|||:, n, x, W, b)[\overrightarrow{T} := \overrightarrow{V}] = (|||:, n, x, W, b[\overrightarrow{T} := \overrightarrow{V}])$, *if* W *is not a parameter*

If there is no ambiguity, the IPASTD is denoted by $a[\overrightarrow{V}]$ *instead of* $a[\overrightarrow{T} := \overrightarrow{V}]$.

For instance, in the library system of Figure A.1, let us instantiate T_m and T_b by $V_m = \{m_1, m_2\}$ and $V_b = \{b_1, b_2, b_3\}$ respectively. We have that $library[V_m, V_b]$ is an IPASTD and an example of a state expression is depicted on Figure A.4. The state consists of two members m_1 and m_2 and three books b_1, b_2 and b_3 where the book b_2 is borrowed by the member m_1. Remark that parameter values should not be empty if we want to be able to instantiate some states.

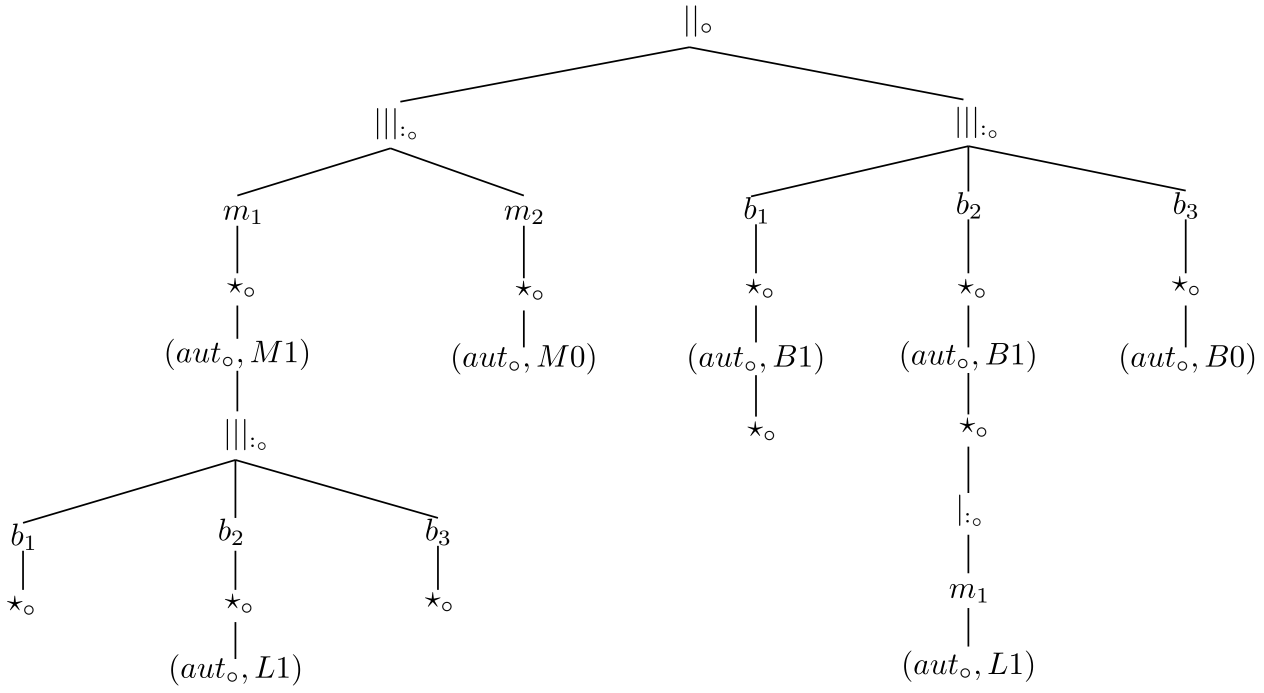


Figure A.4: A state of the library system

If the model contains no free variable (in Automaton ASTD), then an IPASTD $a[\overrightarrow{V}]$ can be seen as a TS. Each state of $a[\overrightarrow{V}]$ is given by an ASTD state expression s and the vector of parameter values $\overrightarrow{V} = (V_1, .., V_k)$, such that s is a well-formed state expression consistent with the instantiated ASTD $a[V_1, .., V_k]$. To be more precise, a state of the TS must also contain the ASTD that the expression s refers to, *i.e.* the PASTD $a[\overrightarrow{T}]$ and the valuation $\overrightarrow{V}$. In [23], states are defined without mentioning any corresponding ASTD as they are implicit. In our case, this will be necessary to compare states from different IPASTD.

Definition 29. *In order to handle PASTD, we make the following modifications to some previous definitions.*

- *In Definition 21: each tuple describing a state is augmented by a PASTD and a valuation. However, for the sake of concision and to keep the same notation, we will denote a tuple representing a state as usual. We denote by $s.pa$ the PASTD associated to a state s and $s.val$ the parameter values.*
- *In Definition 22: if $a[\overrightarrow{V}]$ is an IPASTD, then we define $init(a[\overrightarrow{V}]).pa = a[\overrightarrow{T}]$ and $init(a[\overrightarrow{V}]).val = \overrightarrow{V}$.*
- *In Definition 23: the adaptation for the predicate final is straightforward.*
- *In Definition 25: the transition relation preserves the PASTD and the valuation. If $s \to s'$, then $s.pa = s'.pa$ and $s.val = s'.val$.*

For example, a state s of an Automaton ASTD $a[\overrightarrow{V}]$ will be given by an extended tuple $(\mathsf{aut}_\circ, n, ss, a, \overrightarrow{V})$, or by a simple tuple $(\mathsf{aut}_\circ, n, ss)$ assuming $s.pa = a$ and $s.val = \overrightarrow{V}$.

Note that if a model contains free variables in its Automaton ASTD components, we still need the environment to define a state of a concrete system.

Definition 30. *We denote by $\mathcal{Q}$ the set of IPASTD states s such that s is a well-formed expression with regards to its PASTD $s.pa$ and the valuation $s.val$. In addition to PASTD parameters, $s.pa$ may contain free variables in Automaton ASTD. Formally, $s \in \mathcal{Q}$ iff*

1. $s = (\mathsf{elem}_\circ)$ *and* $s.pa = (\mathsf{elem})$;
2. $s = (\mathsf{aut}_\circ, n, ss)$ *and* $s.pa = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)$ *with* $n \in N$ *and* $ss.pa = \nu(n)$ *and* $ss.val = s.val$ *and* $ss \in \mathcal{Q}$;
3. $s = (\star_\circ, started?, x)$ *and* $s.pa = (\star, n, b)$ *and*
   - $x \neq \bot$ *and* $x.pa = b$ *and* $x.val = s.val$ *and* $x \in \mathcal{Q}$, *or*
   - $x = \bot$ *and* $started? = \text{false}$;
4. $s = (|_\circ, side, x)$ *and* $s.pa = (|, n, l, r)$ *and*
   - $side = \mathsf{left}$ *and* $x.pa = l$ *and* $x.val = s.val$ *and* $x \in \mathcal{Q}$, *or*
   - $side = \mathsf{right}$ *and* $x.pa = r$ *and* $x.val = s.val$ *and* $x \in \mathcal{Q}$, *or*
   - $side = \bot$ *and* $x = \bot$;
5. $s = (||_\circ, s_l, s_r)$ *and* $s.pa = (||, n, \Delta, l, r)$ *and*
   - $s_l.pa = l$ *and* $s_l.val = s.val$ *and* $s_l \in \mathcal{Q}$, *and*
   - $s_r.pa = r$ *and* $s_r.val = s.val$ *and* $s_r \in \mathcal{Q}$;
6. $s = (|:_\circ, v, x)$ *and*
   - $s.pa = (|:, n, y, T_i, b)$ *and* $v \in V_i$ *and* $s.val = \overrightarrow{V}$ *and* $x.pa = b$ *and* $x.val = \overrightarrow{V}$ *and* $x \in \mathcal{Q}$, *where* $T_i \in \overrightarrow{T}$ *is a parameter and* $V_i \in \overrightarrow{V}$ *a value, or*

- $s.pa = (|:, n, y, W, b)$ *and* $v \in W$ *and* $x.pa = b$ *and* $x.val = s.val$ *and* $x \in \mathcal{Q}$*, where* W *is not a parameter, or*
- $s.pa = (|:, n, y, T, b)$ *and* $v = \bot$ *and* $x = \bot$*;*

7. $s = (|||:_\circ, f)$ *and*
   - $s.pa = (|||:, n, y, T_i, b)$ *and* $dom(f) = V_i$ *and* $s.val = \overrightarrow{V}$ *and for all* $ss \in ran(f)$ *we have* $ss.pa = b$ *and* $ss.val = \overrightarrow{V}$ *and* $ss \in \mathcal{Q}$*, where* $T_i \in \overrightarrow{T}$ *is a parameter and* $V_i \in \overrightarrow{V}$ *a value, or*
   - $s.pa = (|||:, n, y, W, b)$ *and* $dom(f) = W$ *and for all* $ss \in ran(f)$ *we have* $ss.pa = b$ *and* $ss.val = s.val$ *and* $ss \in \mathcal{Q}$*, where* W *is not a parameter;*

We are now ready to define the transition system corresponding to an IPASTD. Indeed, as we have seen previously, a state expression of an ASTD is slightly different from a state of a transition system. In the following definition we give a formal definition of IPASTD from a TS viewpoint.

Definition 31. *Let* $a[\overrightarrow{T}]$ *be a PASTD with parameter domains* $\overrightarrow{P}$ *and* $\overrightarrow{V} \subset \overrightarrow{P}$ *a vector of values. If the IPASTD* $a[\overrightarrow{V}]$ *contains no free variables, then we define the corresponding TS* $\mathcal{S}_{a,\overrightarrow{V}} = (Q, \rightarrow)$ *with initial states* $I \subseteq Q$ *as follows:*

- Q *consists of the IPASTD states* $s \in \mathcal{Q}$ *such that* $s.pa = a$ *and* $s.val = \overrightarrow{V}$*;*
- $\rightarrow$ *is the set of all transitions* $s \xrightarrow{\sigma} s'$ *that can be derived in* $a[\overrightarrow{V}]$*, where* s *and* s' *are in* Q*;*
- $I \subseteq Q$*, the set of initial states, is the singleton* $\{init(a[\overrightarrow{V}])\}$*.*

Remark that the function *init* always returns a state of $\mathcal{Q}$ and for all state $s \in \mathcal{Q}$ if $s \rightarrow s'$ then $s' \in \mathcal{Q}$. Thus, every path starting from the initial state possible in the ASTD is modeled in the associated TS.

Lemma 10. *Let* $a[\overrightarrow{T}]$ *be a PASTD. For any parameter value* $\overrightarrow{V}$*,* $init(a[\overrightarrow{V}]) \in \mathcal{Q}$*.*

Lemma 11. *Let* $a[\overrightarrow{T}]$ *be a PASTD and* $\overrightarrow{V}$ *a parameter value. Let* s_1, s_2 *two states such that* $s_1.pa = s_2.pa = a$ *and* $s_1.val = s_2.val = \overrightarrow{V}$*. If* $s_1 \in \mathcal{Q}$ *and* $s_1 \xrightarrow{\sigma,\Gamma} s_2$*, then* $s_2 \in \mathcal{Q}$*.*

Let us now transform PASTD into TS. Intuitively, a parameterized system represents a family of systems, that is a set of systems. In our case, we consider that a parameter of a PASTD can be instantiated by any non-empty finite subset of a (possibly infinite) adequate set of identifiers. We define the global system that includes all possible instantiated systems as follows.

Definition 32. *Let* $a[\overrightarrow{T}]$ *be a PASTD with parameter domains* $\overrightarrow{P}$*. If* $a[\overrightarrow{T}]$ *contains no free variables other than* $\overrightarrow{T}$*, then we define the corresponding TS* $\mathcal{S}_a$ *by the union of all possible instantiations of system:*

$$\mathcal{S}_a = \bigcup_{\overrightarrow{V}} \mathcal{S}_{a,\overrightarrow{V}}$$

where $\overrightarrow{V} = (V_1, ..., V_k)$*,* $\overrightarrow{P} = (P_1, ..., P_k)$ *and* V_i *takes every values in the set of non-empty finite subset of* P_i*. The union of systems is equivalent to a choice between systems. Formally,* $\mathcal{S}_a = (Q, \rightarrow)$ *with initial states* $I \subseteq Q$ *is such that:*

- *Q is the union of the sets of states of each TS;*
- *$\rightarrow$ is the union of the sets of transitions of each TS;*
- *I is the union of the sets of initial states of each TS.*

Remark that $\mathcal{S}_a$ may be an infinite state system, if it is formed by an infinite union of systems. Indeed, if a parameter domain P_i is infinite, then the union over V_i is infinite.
In practice, such global TS that represents parameterized systems are infinite systems. In the example of the library system, the corresponding TS is a infinite system consisting of the union of all possible finite systems instantiated with a natural number of members and of books.

A.4 PASTD are MTS

A.4.1 Defining a quasi-ordering

The theory of Well-Structured Transition Systems is used to check reachability properties in infinite systems like Petri Nets and should be useful to verify other parameterized systems like ours. To apply this method to PASTD, we need to define a qo on its set of states satisfying the monotony condition. Besides, the qo has to be chosen according to the coverability property we want to verify, that is an upward-closed set of states.

Let us consider a PASTD $a[\overrightarrow{T}]$. Intuitively, for a state s of an IPASTD $a[\overrightarrow{V}]$, there is a larger state s' of $a[\overrightarrow{V'}]$ with $\overrightarrow{V} \subseteq \overrightarrow{V'}$, where $\subseteq$ is the point-wise extension of the inclusion relation to vectors, and such that s' can simulate in $a[\overrightarrow{V'}]$ the behavior of s in $a[\overrightarrow{V}]$, *i.e.* every possible execution trace from s is also possible from s'. If the elements of $\overrightarrow{V}$ occur in quantified interleavings, an example of a state s' which is larger than a state s is a state which differs from s only for values in $V'_i \setminus V_i$.

Definition 33. *We define the relation $\sqsubseteq$ on the set $\mathcal{Q}$ of IPASTD states such that $s \sqsubseteq s'$ iff $s.pa = s'.pa$ and $s.val \subseteq s'.val$ and*

1. *$s = (\mathsf{elem}_\circ)$ and $s' = (\mathsf{elem}_\circ)$*
2. *$s = (\mathsf{aut}_\circ, n, ss)$ and $s' = (\mathsf{aut}_\circ, n, ss')$ and $ss \sqsubseteq ss'$*
3. *$s = (\star_\circ, started?, x)$ and $s' = (\star_\circ, started?, x')$ and ($x = x' = \bot$ or $x \sqsubseteq x'$)*
4. *$s = (|_\circ, side, x)$ and $s' = (|_\circ, side, x')$ and ($x = x' = \bot$ or $x \sqsubseteq x'$)*
5. *$s = (||_\circ, s_l, s_r)$ and $s' = (||_\circ, s'_l, s'_r)$ and $s_l \sqsubseteq s'_l$ and $s_r \sqsubseteq s'_r$*
6. *$s = (|:_\circ, v, x)$ and $s' = (|:_\circ, v, x')$ and ($x = x' = \bot$ or $x \sqsubseteq x'$)*
7. *$s = (|||:_\circ, f)$ and $s' = (|||:_\circ, f')$ and $dom(f) \subseteq dom(f')$ and $\forall v \in dom(f) \cdot f(v) \sqsubseteq f'(v)$*

Note that for the last case, $dom(f) \subseteq dom(f')$ is only possible if their respective domains comes from the instantiation of a parameter. Indeed, we require that $s.pa = s'.pa$. Thus, s and s' are incomparable if they do not have the same domain and the quantification set is not a parameter.

Proposition 13. *The relation $\sqsubseteq$ on $\mathcal{Q}$ is a partial ordering.*

Proof. By Definition 33, $\sqsubseteq$ is clearly reflexive, transitive and antisymmetric, as $\subseteq$ is so. $\square$

Unfortunately, monotony does not hold for this partial ordering considering the current transition relation in PASTD. Quantified Interleaving operators do not preserve monotony as every sub-state of quantified interleaving state must synchronize on the final state. Figure A.5 shows an example of PASTD which does not satisfy the monotony. Suppose that the domain of the parameter T is $\mathbb{N}$ and let $V = \{1, 2\} \subseteq V' = \{1, 2, 3\} \subseteq \mathbb{N}$. Let s be a state of $a[V]$ such that both instances of the automaton in $S1$ are in the local state $Q2$. Then, as every instance of the Quantified Interleaving are in final state, the transition F can fire. Consider now a state s' whose ASTD is $a[V']$ and such that the instances 1 and 2 are in $Q2$ but the instance 3 is in $Q1$. According to Definition 33, we have $s \sqsubseteq s'$, as we just add a new instance in an arbitrary local state. However, the transition F cannot be executed from the state s', as the instance 3 is not in a final configuration.

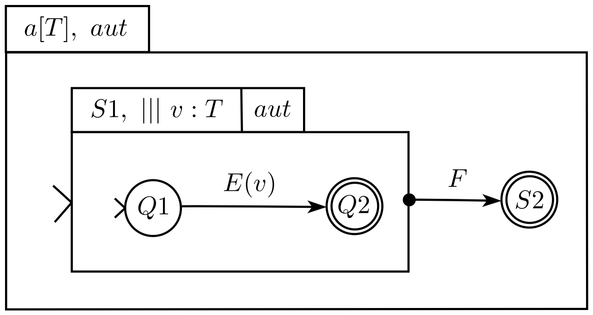


Figure A.5: Example of PASTD

In order to tackle this problem, we change the definition of the predicate *final* such that the following property holds: for a state $s = (|||:_\circ, f)$, if s is final then for each state s' such that $s \sqsubseteq s'$, s' is final too. This allows the transition F, from the previous example, to be able to fire at any moment, thus to preserve the monotony.

Definition 34. *Let $s \in \mathcal{Q}$ be an IPASTD state. The predicate final(s), which determines if s is final in s.pa, is identical to the one in Definition 23 (adapted by Definition 31), except that:*

7. *If $s = (|||:_\circ, f)$ and $s.pa = (|||:, n, x, X, b)[\overrightarrow{T}]$, then*
   - *if X is a parameter, final$(s) \equiv \exists v \in X \cdot$ final$(f(v))$,*
   - *else, final$(s) \equiv \forall v \in X \cdot$ final$(f(v))$.*

Definition 34 is one of the many modifications that can be done to obtain monotony. It means that if the interleaving operator is used with a parameter, then we change the semantics of the quantified interleaving such that we only need one instance to be final for the whole interleaving to be final too. As this can be sometimes inconvenient, we keep the old semantics when X is not a parameter. We could have used a simpler definition where *final*$(s) \equiv$ false or where *final*$(s) \equiv$ true for example. In any way, that modification allows us to prove Lemma 13, which is necessary to show Lemma 14; they are introduced in the sequel.

Lemma 12. *Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter values $\overrightarrow{V}$ and $\overrightarrow{V'}$ such that $\overrightarrow{V} \subseteq \overrightarrow{V'}$, init$(a[\overrightarrow{V}]) \sqsubseteq$ init$(a[\overrightarrow{V'}])$.*

Lemma 13. *For any IPASTD states $s, s' \in \mathcal{Q}$ such that $s \sqsubseteq s'$, if final(s) then final(s').*

Lemma 14. *For any IPASTD states $s_1, s_1', s_2 \in \mathcal{Q}$, any event σ and any environment Γ, if $s_1 \sqsubseteq s_1'$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then there exists s_2' such that $s_2 \sqsubseteq s_2'$ and $s_1' \xrightarrow{\sigma,\Gamma} s_2'$.*

Proofs of Lemmas 12, 13 and 14 are done inductively on the structure of PASTD. Lemmas 12 and 13 give some properties on the function *init* and the predicate *final*. Lemma 14 states a monotony theorem considering an event σ and an environment Γ. The monotony for $\sqsubseteq$ is a simple corollary of this lemma. Proofs are provided in Appendix B.2.

Theorem 7. *Let $a[\overrightarrow{T}]$ be a PASTD without free variables in events, with $\mathcal{S}_a$ the corresponding TS. $\mathcal{S}_a$ is monotone wrt $\sqsubseteq$.*

Proof. Let s_1, s_1', s_2 states of $\mathcal{S}_a$ such that $s_1 \sqsubseteq s_1'$ and $s_1 \to s_2$. There is an event σ such that $s_1 \xrightarrow{\sigma} s_2$. By the only inference rule env, we have $s_1 \xrightarrow{\sigma,(\!|\,|\!)} s_2$. Then, by Lemma 14 there exists $s_2' \in \mathcal{Q}$ such that $s_2 \sqsubseteq s_2'$ and $s_1' \xrightarrow{\sigma,(\!|\,|\!)} s_2'$. Thus, by the rule env, we have $s_1' \xrightarrow{\sigma} s_2'$. As s_2' is a state of $\mathcal{S}_a$ too, we can conclude. $\square$

Nevertheless, Definition 34 could be problematic in some cases. For example, recall the example of the library system of Figure A.1, where a member must complete, *i.e.* return, all her loans in order to leave the library system. If we consider the previous definition, she would be able to leave whenever she has returned one of her loans without completing the other loan processes. This scenario could be an issue of that new definition, but we propose a solution to solve the problem in that case. Suppose that the number of loans is limited to 2 books per member. Then, we can modify the ASTD of the member process to obtain the one of Figure A.6, where symbol "_" in events means that the symbol can be replaced by any value. Indeed, we add a new Automaton ASTD in synchronization with the Quantified Interleaving in order to be able to specify a final state for the ASTD $M1$. Intuitively, this automaton counts the number of loans and allows to terminate only if this number is equal to zero. In our case, the member will not be allowed to leave if she has not returned all her loans. This principle can be generalized to every ASTD specification where we can limit the number of processes in the Quantified Interleaving, because the counting automaton must be finite state.

A.4.2 Symmetries

The notion of *symmetry* is an important topic in the context of model checking. In [17], the authors show several way to exploit isomorphisms in systems in order to simplify the verification process. We use here a similar approach in PASTD and we will see that the detection of symmetries is essential in our case.

In PASTD, we are interested in the symmetries appearing in parameter domains. Indeed, the name of an instance has no importance in the system and can always be renamed by another element of the parameter domain. We consider that two states are equivalent if they are syntactically equivalent under some specific rename function that renames all occurrences of a parameter element by another one. Let us define formally those rename functions.

Definition 35 (Rename Function)**.** *Let $P_1, P_2, ..., P_k$ a set of parameter domains. A* rename function *ξ for the set of IPASTD states $\mathcal{Q}$ is defined by a set of permutations $\rho_1, \rho_2, ..., \rho_k$ on $P_1, P_2, ..., P_k$ respectively such that:*

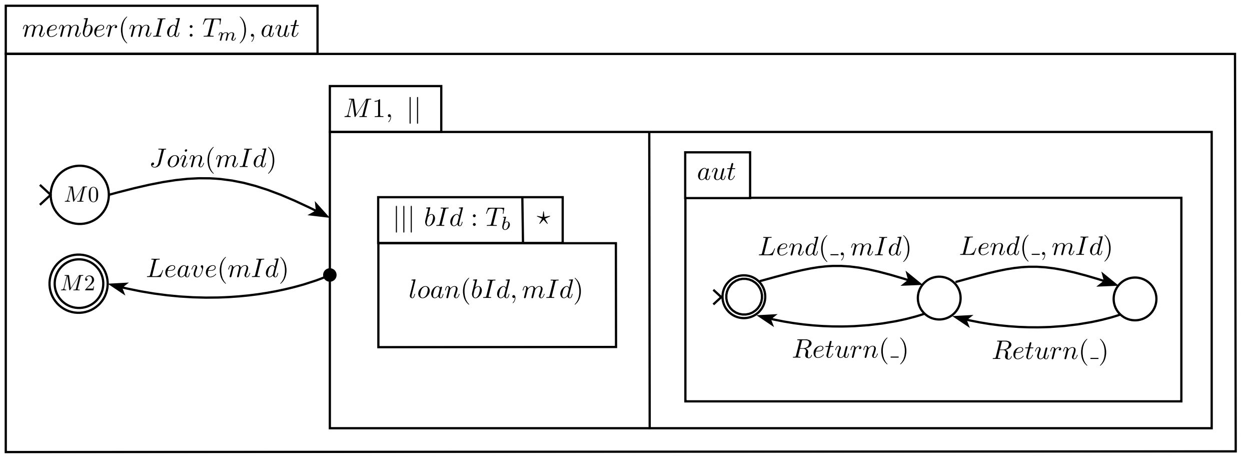


Figure A.6: Modification of the member process

1. $\xi((\mathsf{elem}_\circ)) = (\mathsf{elem}_\circ)$
2. $\xi((\mathsf{aut}_\circ, n, s)) = (\mathsf{aut}_\circ, n, \xi(s))$
3. $\xi((\star_\circ, started?, s)) = \begin{cases} (\star_\circ, \text{false}, \bot) & \textit{if } started? = \text{false} \\ (\star_\circ, \text{true}, \xi(s)) & \textit{else} \end{cases}$
4. $\xi((|_\circ, side, x)) = \begin{cases} (|_\circ, \bot, \bot) & \textit{if } side = \bot \\ (|_\circ, side, \xi(x)) & \textit{else} \end{cases}$
5. $\xi((||_\circ, s_l, s_r)) = (||_\circ, \xi(s_l), \xi(s_r))$
6. $\xi((|:_\circ, v, x)) = \begin{cases} (|:_\circ, \bot, \bot) & \textit{if } v = \bot \\ (|:_\circ, \rho_i(v), \xi(x)) & \textit{if } \exists i \cdot v \in P_i \\ (|:_\circ, v, \xi(x)) & \textit{else} \end{cases}$
7. $\xi((|||:_\circ, f)) = \begin{cases} (|||:_\circ, f') & \textit{if } \exists i \cdot dom(f) \subseteq P_i \\ (|||:_\circ, f'') & \textit{else} \end{cases}$
   where $dom(f') = \rho_i(dom(f))$ *and for all* $v \in dom(f)$ *we have* $f'(\rho_i(v)) = \xi(f(v))$*, and* $dom(f'') = dom(f)$ *and for all* $v \in dom(f)$ *we have* $f''(v) = \xi(f(v))$

and $\xi(s).pa = s.pa$ *and* $\xi(s).val = (\rho_1(V_1), ...\rho_k(V_k))$*, where* $s.val = (V_1, ..., V_k)$*.*

Now we can define a quasi-ordering on $\mathcal{Q}$ regarding the equivalence relation induced by the rename functions.

Definition 36. *We define the relation* $\preceq$ *on* $\mathcal{Q}$ *by:*

$$s \preceq s' \iff \exists \xi \text{ a rename function s.t. } s \sqsubseteq \xi(s')$$

Proposition 14. *The relation* $\preceq$ *on* $\mathcal{Q}$ *is a quasi-ordering.*

Proof. The relation $\preceq$ is reflexive because the identity is a rename function and $\sqsubseteq$ is reflexive. The relation $\preceq$ is transitive because the composition of rename functions is a rename function and $\sqsubseteq$ is transitive. $\square$

The notion of renaming is useful to factorize the state space of a PASTD. In fact, it is necessary in order to satisfy the finite pred-basis property needed in WSTS. Consider the PASTD of Figure A.5. Let $s = (\mathsf{aut}_\circ, S2, (\mathsf{elem}_\circ))$ be the state where we are at the local state $S2$. A predecessor of s is a state $s' = (\mathsf{aut}_\circ, S1, (|||{:}_\circ, \{1 \mapsto (\mathsf{aut}_\circ, Q2, (\mathsf{elem}_\circ))\}))$ where the instance 1 of the Quantified Interleaving in the local state $S1$. But $s'' = (\mathsf{aut}_\circ, S1, (|||{:}_\circ, \{2 \mapsto (\mathsf{aut}_\circ, Q2, (\mathsf{elem}_\circ))\}))$ is also a predecessor of s. However, s' and s'' are incomparable and have no lower bound. Thus the pred-basis of s includes s' and s''. As the parameter domain may be infinite, we can conclude that s has no finite pred-basis if we consider the qo $\sqsubseteq$. This is not the case with $\preceq$, because $s' \preceq s''$ (take a rename function defined by a permutation ρ on $\mathbb{N}$ such that $\rho(1) = 2$ and $\rho(2) = 1$).

The following three lemmas are very similar to those from the previous section and allow to do the proof of monotony. Their proofs have the same structure and are detailed in Appendix B.3.

Lemma 15. *Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter value $\overrightarrow{V}$ and any rename function ξ defined by permutations $\rho_1, ..., \rho_k$, we have*

$$\xi(init(a[\overrightarrow{V}])) = init(a[\rho_1(V_1), ..., \rho_k(V_k)])$$

Lemma 16. *For any $s \in \mathcal{Q}$ and for any rename function ξ, if final(s), then final$(\xi(s))$.*

Lemma 17. *Let $P_1, ..., P_k$ a set of parameter domains. For any states $s_1, s_2 \in \mathcal{Q}$, any rename function ξ defined by permutations $\rho_1, ..., \rho_k$ on $P_1, ..., P_k$ respectively, any event σ and any environment $\Gamma = (\![x_1, ..., x_n := v_1, ..., v_n]\!)$, if $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$, where $\sigma' = \sigma[v_1, ..., v_n := v'_1, ..., v'_n]$ and $\Gamma' = (\![x_1, ..., x_n := v'_1, ..., v'_n]\!)$ with $v'_i = \rho_j(v_i)$ if $\exists j \cdot v_i \in P_j$ or $v'_i = v_i$ else.*

We can now state the final monotony theorem for the qo $\preceq$. Its proof results from Lemma 17 and the monotony for $\sqsubseteq$.

Theorem 8. *Let $a[\overrightarrow{T}]$ be a PASTD without free variables in events, with $\mathcal{S}_a$ the corresponding TS. $\mathcal{S}_a$ is monotone wrt $\preceq$.*

Proof. Let s_1, s_2, s'_1 three states of $\mathcal{S}_a$ such that $s_1 \preceq s'_1$ and $s_1 \to s_2$. There is an event σ such that $s_1 \xrightarrow{\sigma} s_2$ and by the rule env we have $s_1 \xrightarrow{\sigma,(\![]\!)} s_2$. By Definition 36, there exists a rename function ξ such that $s_1 \sqsubseteq s''_1$ with $s''_1 = \xi(s'_1)$. And by Lemma 14, let s''_2 such that $s_2 \sqsubseteq s''_2$ and $s''_1 \xrightarrow{\sigma,(\![]\!)} s''_2$. Let $s'_2 = \xi^{-1}(s''_2)$, then we have $s'_1 = \xi^{-1}(s''_1) \xrightarrow{\sigma',(\![]\!)} \xi^{-1}(s''_2) = s'_2$ by Lemma 17, with σ' the renaming of σ. As $s_2 \sqsubseteq s''_2$, then $s_2 \preceq \xi^{-1}(s''_2) = s'_2$. The inference rule env gives $s'_1 \xrightarrow{\sigma'} s'_2$, thus $s'_1 \to s'_2$. We have s'_2 state of $\mathcal{S}_a$, thus $\mathcal{S}_a$ is monotone. □

Intuitively, renaming preserves monotony because the roles of identifiers from parameter domains are symmetrical in the system, as proved by Lemma 17. This is the reason why we do not allow the use of an element or a subset of a parameter domain as a constant in a PASTD specification. For example, the specification in Figure A.7 is invalid because it uses the event $E(1)$ in a transition. It is easy to see that ASTD $main[\{1\}]$ and $main[\{2\}]$ have different behaviors. Note that the event $E(1)$ from the right-hand side sub-ASTD is not affected by any rename function, as it is not part of any state expression.

Furthermore, the reachability property must satisfy the same constraint of symmetry, that is the set R must be upward-closed with regards to the qo $\preceq$. This should not be an issue,

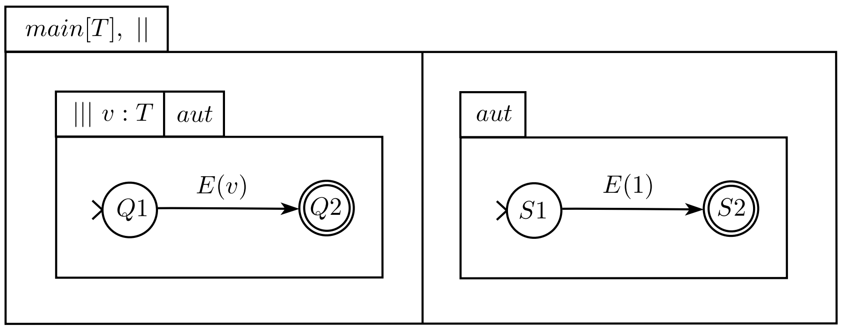


Figure A.7: Example of invalid PASTD

since, in most cases, properties do not deal with specific instances of a set of elements. Take for example the PASTD of Figure A.5, where parameter T has domain $\mathbb{N}$. Determining if the state $s = (\mathsf{aut}_\circ, S1, (|||:_\circ, \{1 \mapsto (\mathsf{aut}_\circ, Q2, (\mathsf{elem}_\circ))\}))$ is reachable is equivalent to determining if any state of kind $(\mathsf{aut}_\circ, S1, (|||:_\circ, \{id \mapsto (\mathsf{aut}_\circ, Q2, (\mathsf{elem}_\circ))\}))$, where id is an element of $\mathbb{N}$, is reachable.

A.5 Extending the state space

In order to satisfy the *backward-downward monotony*, we need to make some modifications to our systems. Indeed, with the current definition of IPASTD states (Definition 30), the bdm property does not hold and thus a state may not have a finite pred-basis.

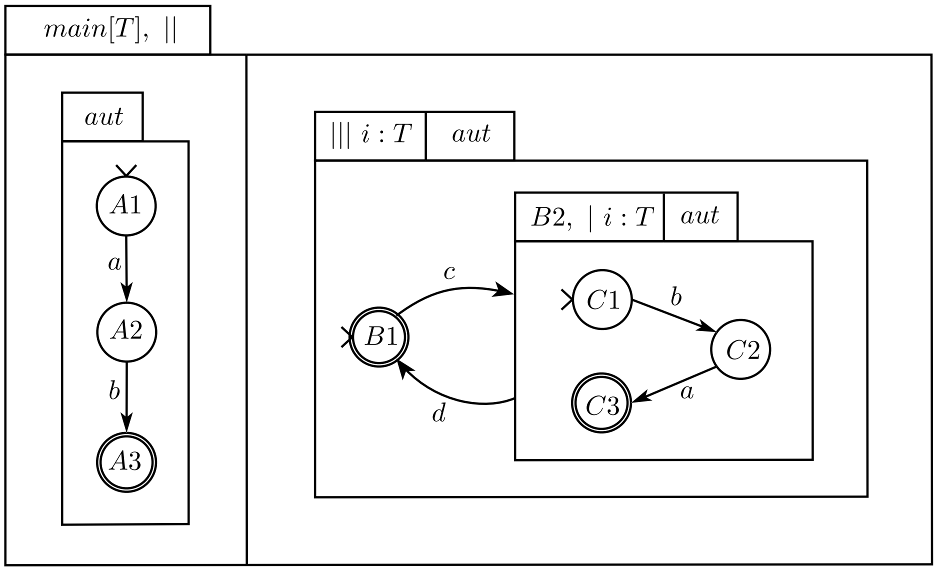


Figure A.8: Example of PASTD

Consider for example the PASTD of Figure A.8, with $\mathbb{N}$ as the parameter domain of T, and a state $s = (||_\circ, (\mathsf{aut}_\circ, A3, (\mathsf{elem}_\circ)), (|||:_\circ, f))$, where the function for the quantified interleaving is defined by $f = \{1 \mapsto (\mathsf{aut}_\circ, B2, (|:_\circ, 1, (\mathsf{aut}_\circ, C3, (\mathsf{elem}_\circ))))\}$, and with $s.pa = main$ and $s.val =$

({1}). Figure A.9 shows some elements of the set $\uparrow Pred(\uparrow s)$, which are not comparable. And as we can find an infinite number of such states, we can deduce that there is no finite basis for that set. The transition relation does not satisfy the bdm. This is because each instance appearing in the quantified choice has to appear in the quantified interleaving too. (Note that it is not the case if the set of states is well-quasi-ordered.)

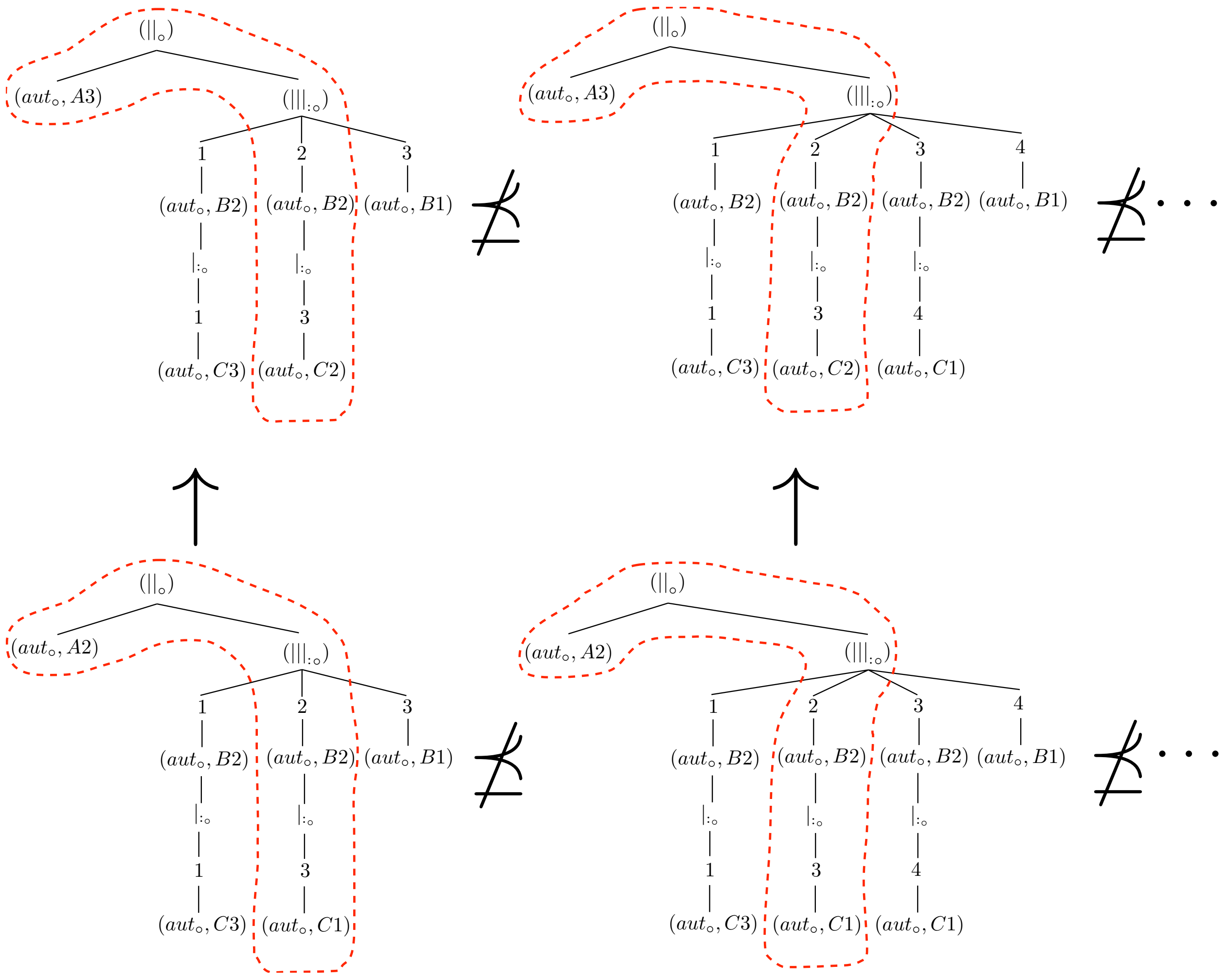


Figure A.9: Example of a set of predecessors without a common basis

However, we can notice that the central part of the transition (encircled by dotted lines on the figure) can be localized in the state structure and is the same one. Thus, we should be able to construct a lower bound for that set of predecessors, which would correspond to some state $s' = (||_\circ, (\mathsf{aut}_\circ, A2, (\mathsf{elem}_\circ)), (|||:_\circ, f'))$, where the function f' is defined by $f' = \{1 \mapsto (\mathsf{aut}_\circ, B2, (|:_\circ, 1, (\mathsf{aut}_\circ, C3, (\mathsf{elem}_\circ)))), 2 \mapsto (\mathsf{aut}_\circ, B2, (|:_\circ, 3, (\mathsf{aut}_\circ, C1, (\mathsf{elem}_\circ))))\}$, and with $s.pa = main$ and $s.val = (\{1, 2, 3\})$. But, as $dom(f') \neq \{1, 2, 3\}$, s' is not a well-formed state according to Definition 30.

To overcome this issue, we augment the set of states to consider in PASTD systems by adding some specific abstract states. Intuitively, an abstract state is a state where the local configurations of some instances are unknown. Such an abstract state s represents the set of all concrete states where the quantified interleavings are completely instantiated but include the instances appearing in s within their configurations. Formally, the new set of IPASTD states is defined by similarly as Definition 30 except for the last case.

Definition 37. *We denote by $\mathcal{Q}^\sharp$ the set of IPASTD states such that $s \in \mathcal{Q}^\sharp$ iff either it verifies one of the cases from 1 to 6 of Definition 30 or,*

7. *$s = (|||:_\circ, f)$ and*
   - *$s.pa = (|||:, n, y, T_i, b)$ and $dom(f) \subseteq V_i$ and $dom(f) \neq \emptyset$ and $s.val = \overrightarrow{V}$ and for all $ss \in ran(f)$ we have $ss.pa = b$ and $ss.val = \overrightarrow{V}$ and $ss \in \mathcal{Q}^\sharp$, where $T_i \in \overrightarrow{T}$ is a parameter and $V_i \in \overrightarrow{V}$ a value, or*
   - *$s.pa = (|||:, n, y, W, b)$ and $dom(f) = W$ and for all $ss \in ran(f)$ we have $ss.pa = b$ and $ss.val = s.val$ and $ss \in \mathcal{Q}^\sharp$, where W is not a parameter;*

The difference with Definition 30 is that the quantified interleaving case $s = (|||:_\circ, f)$ with $s.pa = (|||:, n, y, T_i, b)$, does not require the domain of f to match the entire set of values V_i but only a non-empty subset, where T_i is a parameter. Clearly, $\mathcal{Q}^\sharp$ is a superset of $\mathcal{Q}$.

We can extend the quasi-orderings $\sqsubseteq$ and $\preceq$ of Definition 33 and 36 to the set $\mathcal{Q}^\sharp$. Indeed, Definitions 33 and 35 can be applied to the set $\mathcal{Q}^\sharp$, as the quantified interleaving cases $(|||:_\circ, f)$ of both definitions refer to the domain of the function f. We keep the same notation for the quasi-ordering extended to $\mathcal{Q}^\sharp$.

Proposition 15. *The relations $\sqsubseteq$ and $\preceq$ are quasi-ordering on $\mathcal{Q}^\sharp$.*

Considering the new set of states $\mathcal{Q}^\sharp$, the TS associated to an IPASTD has its set of states augmented as well as the transition relation. First, let us introduce a function *Abinit* which characterizes the set of abstract initial states.

Definition 38. *Let $a[\overrightarrow{T}]$ be a PASTD and $\overrightarrow{V}$ a parameter value. We define the set $Abinit(a[\overrightarrow{V}])$ by:*

$$Abinit(a[\overrightarrow{V}]) = \{s \in \mathcal{Q}^\sharp \mid s \sqsubseteq init(a[\overrightarrow{V}])\}$$

For any concrete initial state $init(a[\overrightarrow{V}])$, we define the set of its corresponding abstract states $Abinit(a[\overrightarrow{V}])$ by the abstract states from $\mathcal{Q}^\sharp$ whose branches can be completed to obtain $init(a[\overrightarrow{V}])$.

Furthermore, we replace some transition rules from Definition 25 to deal with abstract initial states. Each condition involving an initial state $init(a)$ is simply replaced by a condition that uses the function *Abinit*.

Definition 39. *The operational semantics of PASTD is given by Definition 25 except for the following cases.*

- *If $a[\overrightarrow{V}] = (\mathsf{aut}, n, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{V}]$, the rule aut_1 is replaced by the following rule aut_{1a}:*

$$\mathsf{aut}_{1a}\ \frac{\delta(n_1, n_2, \sigma', final?) \qquad final? \Rightarrow final(s) \wedge \sigma'[\Gamma] = \sigma \qquad \chi}{(\mathsf{aut}_\circ, n_1, s) \xrightarrow{\sigma,\Gamma} (\mathsf{aut}_\circ, n_2, s')}$$

where $\chi = s' \in Abinit(\nu(n_2)[\overrightarrow{V}])$

- *If* $a[\overrightarrow{V}] = (\star, n, b[\overrightarrow{V}])$*, the rule* $\star_1$ *is replaced by the following rule* $\star_{1a}$*:*

$$\star_{1a}\ \frac{(\mathit{final}(s) \lor \neg \mathit{started?}) \qquad q \xrightarrow{\sigma,\Gamma}_{b[\overrightarrow{V}]} s' \qquad q \in Abinit(b[\overrightarrow{V}])}{(\star_\circ, \mathit{started?}, s) \xrightarrow{\sigma,\Gamma} (\star_\circ, \mathrm{true}, s')}$$

- *If* $a[\overrightarrow{V}] = (|, n, l[\overrightarrow{V}], r[\overrightarrow{V}])$*, rules* $|_1$ *and* $|_2$ *are respectively replaced by rules* $|_{1a}$ *and* $|_{2a}$*:*

$$|_{1a}\ \frac{q \xrightarrow{\sigma,\Gamma}_{l[\overrightarrow{V}]} s' \qquad q \in Abinit(l[\overrightarrow{V}])}{(|_\circ, \perp, \perp) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathit{left}, s')}$$

$$|_{2a}\ \frac{q \xrightarrow{\sigma,\Gamma}_{r[\overrightarrow{V}]} s' \qquad q \in Abinit(r[\overrightarrow{V}])}{(|_\circ, \perp, \perp) \xrightarrow{\sigma,\Gamma} (|_\circ, \mathit{right}, s')}$$

- *If* $a[\overrightarrow{V}] = (|:, n, x, T, b[\overrightarrow{V}])$*, the rule* $|:_1$ *is replaced by the following rule:*

$$|:_{1a}\ \frac{q \xrightarrow{\sigma,([x:=v])\lhd\Gamma}_{b[\overrightarrow{V}]} s' \qquad q \in Abinit(b[\overrightarrow{V}]) \qquad v \in T}{(|:_\circ, \perp, \perp) \xrightarrow{\sigma,\Gamma} (|:_\circ, v, s')}$$

Note that for each transition $s \xrightarrow{\sigma,\Gamma} s'$*, we still have implicitly that* $s.pa = s'.pa$ *and* $s.val = s'.val$*. Moreover, these rules are specific to PASTD because of the definition of the function Abinit.*

The other rules from Definition 25 take into account those abstract states as the rule for the quantified interleaving operator applies to a function f with any domain of definition.

Similarly, we redefine the TS associated to an IPASTD by adding the abstract initial states.

Definition 40. *Let* $a[\overrightarrow{T}]$ *be a PASTD with parameter domains* $\overrightarrow{P}$ *and* $\overrightarrow{V} \subset \overrightarrow{P}$ *a vector of values. If the IPASTD* $a[\overrightarrow{V}]$ *contains no free variables, then we define the corresponding TS* $\mathcal{S}^\sharp_{a,\overrightarrow{V}} = (Q, \rightarrow)$ *with abstract states as follows:*

- Q *consists of the IPASTD states* $s \in \mathcal{Q}^\sharp$ *such that* $s.pa = a$ *and* $s.val = \overrightarrow{V}$*;*
- $\rightarrow$ *is the set of all transitions* $s \xrightarrow{\sigma} s'$ *that can be derived in* $a[\overrightarrow{V}]$ *following Definition 39, where* s *and* s' *are in* Q*;*
- $I \subseteq Q$*, the set of initial states, is the set* $\{q \mid q \in Abinit(a[\overrightarrow{V}])\}$*.*

By convention, if $a[\overrightarrow{V}]$ is an IPASTD and $\mathcal{S}_{a,\overrightarrow{V}}$ the TS derived from Definition 30, then we denote by $\mathcal{S}^\sharp_{a,\overrightarrow{V}}$ the TS derived from the previous definition. For a PASTD $a[\overrightarrow{T}]$ the TS corresponding to the definition with abstract states is given as follows (similar to Definition 32).

Definition 41. *Let $a[\overrightarrow{T}]$ be a PASTD with parameter domains $\overrightarrow{P}$. If $a[\overrightarrow{T}]$ contains no free variables other than $\overrightarrow{T}$, then we define the corresponding TS $\mathcal{S}^\sharp_a$ by the union of all possible instantiations of system:*

$$\mathcal{S}^\sharp_a = \bigcup_{\overrightarrow{V}} \mathcal{S}^\sharp_{a,\overrightarrow{V}}$$

where $\overrightarrow{V} = (V_1, ..., V_k)$, $\overrightarrow{P} = (P_1, ..., P_k)$ and V_i takes every values in the set of non-empty finite subset of P_i.

Compared to the TS $\mathcal{S}_a$, the new system $\mathcal{S}^\sharp_a$ has a larger set of states and of transitions.

Now, we must ensure that the extended quasi-orderings preserves the monotony of $\mathcal{S}^\sharp_a$. The proof is similar to the one for Theorem 7 and is given in Appendix B.4.

Theorem 9. *Let $a[\overrightarrow{T}]$ be a PASTD without free variables in events, with $\mathcal{S}^\sharp_a$ the corresponding TS from Definition 41. $\mathcal{S}^\sharp_a$ is monotone wrt $\sqsubseteq$ and $\preceq$.*

If we change the TS to consider in the procedure, we need to show that the verification is still correct. Consider a PASTD $a[\overrightarrow{T}]$, the TS $\mathcal{S}_a = (Q, \rightarrow)$ with initial states $I \subseteq Q$, which is derived from Definition 32, and the TS $\mathcal{S}^\sharp_a = (Q', \rightarrow')$ with initial states $I' \subseteq Q'$ from Definition 41. Suppose that we want to determine if $s \in Q$ is coverable in $a[\overrightarrow{T}]$. We can replace $\mathcal{S}_a$ by $\mathcal{S}^\sharp_a$ in the verification procedure only if they are equivalent with regards to the coverability problem, *i.e.* $cov(\mathcal{S}_a, I, s) \iff cov(\mathcal{S}^\sharp_a, I', s)$.

Lemma 18. *Let $a[\overrightarrow{T}]$ be a PASTD, $\mathcal{S}_a = (Q, \rightarrow)$ the TS derived from Definition 32 with initial states I, and $\mathcal{S}^\sharp_a = (Q', \rightarrow')$ the TS from Definition 41 with initial states I'. Then, for all path $s'_0 \rightarrow' ... \rightarrow' s'_n$ in $\mathcal{S}^\sharp_a$ with $s'_0 \in I'$, there exists a path $s_0 \rightarrow ... \rightarrow s_n$ in $\mathcal{S}_a$ with $s_0 \in I$ such that $s'_i \leq s_i$ for all $i \leq n$.*

We can now state the theorem that support our extension of state space.

Theorem 10. *Let $a[\overrightarrow{T}]$ be a PASTD, $\mathcal{S}_a = (Q, \rightarrow)$ the TS derived from Definition 32 with initial states I, and $\mathcal{S}^\sharp_a = (Q', \rightarrow')$ the TS from Definition 41 with initial states I'. Let $s \in Q$ a state. Then, $cov(\mathcal{S}_a, I, s) \iff cov(\mathcal{S}^\sharp_a, I', s)$.*

Proof. $cov(\mathcal{S}_a, I, s) \implies cov(\mathcal{S}^\sharp_a, I', s)$ is trivial. By Lemma 18, $cov(\mathcal{S}_a, I, s) \impliedby cov(\mathcal{S}^\sharp_a, I', s)$. □

Remark that some interesting reachability properties in $\mathcal{Q}^\sharp$ cannot be represented in $\mathcal{Q}$. Take for instance the example of Figure A.8. Let R be the set of states greater than the state $(||_\circ, (\mathsf{aut}_\circ, A3, (\mathsf{elem}_\circ)), (|||:_\circ, f))$, such that $f = \{i \mapsto (\mathsf{aut}_\circ, B2, (|:_\circ, j, (\mathsf{aut}_\circ, C2, (\mathsf{elem}_\circ))))\}$, for all $i \neq j$. We have seen in Figure A.9 that R has no finite basis in $\mathcal{Q}$ but can be represented by a finite basis in $\mathcal{Q}^\sharp$. We can then verify the coverabilty of a finite set of states in $\mathcal{Q}^\sharp$. From now, we will consider that coverability problems are directly specified in $\mathcal{Q}^\sharp$.

A.6 PASTD are RMTS

In this section we use RMTS to prove that PASTD have effective pred-basis. Because we use a quasi-ordering on the state space of PASTD, we cannot say that PASTD are RMTS as it require a partial ordering. However, it is natural to consider the quotient set of the state space instead of the entire space when studying systems like PASTD. Exploiting the symmetries in systems in order to simplify a verification process is usual in the context of formal verification [17]. In our case the symmetries entailed by the equivalence relation allow us to consider the coverability problem on the quotient space.

For a PASTD $a[\overrightarrow{T}]$ with $\mathcal{S}^\sharp_a = (Q, \to)$, $\mathcal{T}_a/\!\sim$ is the quotient set of $\mathcal{T}_a$ and for any relation (transition relation or ordering in our case) on $\mathcal{T}_a$, we use the same notation for the corresponding one on $\mathcal{T}_a/\!\sim$. More formally, $\tilde{s_1} \preceq \tilde{s_2}$ if there is $s_1 \in \tilde{s_1}$ and $s_2 \in \tilde{s_2}$ such that $s_1 \preceq s_2$, and $\tilde{s_1} \to \tilde{s_2}$ if there is $s_1 \in \tilde{s_1}$ and $s_2 \in \tilde{s_2}$ such that $s_1 \to s_2$. We denote by $\tilde{\mathcal{S}}_a = (\mathcal{T}_a/\!\sim, \to)$ the quotient system.

Proposition 16. *Let $a[\overrightarrow{T}]$ be a PASTD and $\mathcal{S}^\sharp_a = (Q, \to)$. Then, $(\mathcal{T}_a, \to, \sim)$ and $(\mathcal{T}_a/\!\sim, \to, \preceq)$ are both monotone.*

Proof. The proof of monotony of $(\mathcal{T}_a, \to, \sim)$ is similar to the one for $(\mathcal{T}_a, \to, \preceq)$. Let us prove that $(\mathcal{T}_a/\!\sim, \to, \preceq)$ is monotone. Let $\tilde{s_1}, \tilde{s'_1}, \tilde{s_2} \in \mathcal{T}_a/\!\sim$ such that $\tilde{s_1} \preceq \tilde{s'_1}$ and $\tilde{s_1} \to \tilde{s_2}$. Then, there exist $s_1, q_1 \in \tilde{s_1}$ and $s_2 \in \tilde{s_2}$ and $s'_1 \in \tilde{s'_1}$ such that $q_1 \preceq s'_1$ and $s_1 \to s_2$. By monotony of $(\mathcal{T}_a, \to, \sim)$, there exists $q_2 \in \tilde{s_2}$ such that $q_1 \to q_2$. Thus, by monotony of $(\mathcal{T}_a, \to, \preceq)$, there exists $s'_2 \in \mathcal{T}_a$ such that $q_2 \preceq s'_2$ and $s'_1 \to s'_2$. Let $\tilde{s'_2}$ be the equivalence class of s'_2. Then, we have $\tilde{s_2} \preceq \tilde{s'_2}$ and $\tilde{s'_1} \to \tilde{s'_2}$. □

The monotony of $(\mathcal{T}_a, \to, \sim)$ allows to prove that the coverability problem in $\mathcal{S}^\sharp_a$ can be reduced to the coverability problem in $\tilde{\mathcal{S}}_a$.

Theorem 11. *Let $a[\overrightarrow{T}]$ be a PASTD, $\mathcal{S}^\sharp_a = (\mathcal{T}_a, \to, \preceq)$ its corresponding OTS and $\tilde{\mathcal{S}}_a = (\mathcal{T}_a/\!\sim, \to, \preceq)$ its quotient OTS. Let $I \subseteq \mathcal{T}_a$, $s \in \mathcal{T}_a$, $\tilde{I} = \{\tilde{i} \in \mathcal{T}_a/\!\sim \mid \exists i \in I \cdot i \in \tilde{i}\}$ and $\tilde{s}$ the equivalence class of s. Then, $cov(\mathcal{S}^\sharp_a, I, s) \iff cov(\tilde{\mathcal{S}}_a, \tilde{I}, \tilde{s})$.*

Proof. Let us prove that $\exists i \xrightarrow{*} s' \cdot s \preceq s' \wedge i \in I \Leftrightarrow \exists \tilde{i} \xrightarrow{*} \tilde{s'} \cdot \tilde{s} \preceq \tilde{s'} \wedge \tilde{i} \in \tilde{I}$. $\Rightarrow$ is trivial. $\Leftarrow$ is true because $(\mathcal{T}_a, \to, \sim)$ is monotone (see Proposition 16). □

From now on, for a PASTD $a[\overrightarrow{T}]$ we will study the coverability problem on the quotient MTS $\tilde{\mathcal{S}}^\sharp_a = (\mathcal{T}_a/\!\sim, \to, \preceq)$. The objective is to find a RMTS for $a[\overrightarrow{T}]$. Let us define an adequate rank function.

Definition 42. *We define a function $\gamma : \mathcal{Q}^\sharp \to \mathbb{N}^k$ on the set of IPASTD states by $\gamma(s) = (|V_1|, ..., |V_k|)$ for all $s \in \mathcal{Q}^\sharp$ with $s.val = (V_1, ..., V_k)$.*

The function γ gives for each state s the number of instances of each type that appear within the description of s. For instance, if s is the state of Figure A.4 from the library example, then $\gamma(s) = (2, 3)$ as $s.val = (\{m_1, m_2\}, \{b_1, b_2, b_3\})$. The state describes a configuration with 2 members and 3 books.

Clearly, for all $s_1, s_2 \in \tilde{s} \in \mathcal{Q}^\sharp/\!\sim$, $\gamma(s_1) = \gamma(s_2)$. Thus, we can extend the rank function to the equivalence classes $\gamma : \mathcal{Q}^\sharp/\!\sim \to \mathbb{N}^k$ as $\gamma(\tilde{s}) = \gamma(s)$ for any $s \in \tilde{s}$. The function has the following property with regards to transitions.

Proposition 17. *For all $\tilde{s_1}, \tilde{s_2} \in \mathcal{Q}^\sharp/\!\sim$, such that $\tilde{s_1} \to \tilde{s_2}$, we have $\gamma(\tilde{s_1}) = \gamma(\tilde{s_2})$.*

Proof. By Definitions 39 and 42. □

Now, for each PASTD, we can determine a $c \in \mathbb{N}^n$ which is a bound satisfying Condition 4 of Definition 14.

Definition 43 (Maximum transition size)**.** *Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD. The function μ, which gives for each PASTD expression F a vector of natural $c \in \mathbb{N}^n$ called* maximum transition size, *is defined recursively as follows.*

1. $\mu((\mathsf{elem})[\overrightarrow{T}]) = (0, ..., 0)$
2. $\mu((\mathsf{aut}, n, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]) = sup(\{\mu(\nu(n')) \mid n' \in N\})$
3. $\mu((\star, n, b)[\overrightarrow{T}]) = \mu(b[\overrightarrow{T}])$
4. $\mu((|, n, l, r)[\overrightarrow{T}]) = sup(\{\mu(l[\overrightarrow{T}]), \mu(r[\overrightarrow{T}])\})$
5. $\mu((||, n, \Delta, l, r)[\overrightarrow{T}]) = \mu(l[\overrightarrow{T}]) + \mu(r[\overrightarrow{T}])$
6. $\mu((|:, n, x, X, b)[\overrightarrow{T}]) = \begin{cases} \mu(b[\overrightarrow{T}]) & \textit{if } X \textit{ is not a parameter} \\ \mu(b[\overrightarrow{T}]) + (u_1, ..., u_k) & \textit{if } X = T_i \\ & \quad \textit{where } u_i = 1 \textit{ and} \\ & \quad u_j = 0 \textit{ for all } i \neq j \end{cases}$
7. $\mu((|||:, n, x, X, b)[\overrightarrow{T}]) = \begin{cases} |X| \cdot \mu(b[\overrightarrow{T}]) & \textit{if } X \textit{ is not a parameter} \\ \mu(b[\overrightarrow{T}]) + (u_1, ..., u_k) & \textit{if } X = T_i \\ & \quad \textit{where } u_i = 1 \textit{ and} \\ & \quad u_j = 0 \textit{ for all } i \neq j \end{cases}$

Note that for all $C \subset \mathbb{N}^k$, we denote by $sup(C)$ the supremum of the set C in $\mathbb{N}^k$.

For instance, consider the PASTD of the library illustrated in Figure A.1. By Definition 43, we have $\mu(F) = (2, 2)$ (remark that each parameter appears twice in the tree). It means that for each transition in the system, a maximum of 2 members and 2 books will be actually involved.

The function μ determines the maximum number of instances of each type that are involved in a transition. Intuitively, we count one instance for each quantified operator associated to the corresponding parameter. Remark that there will be at least one instance for each parameter that is used in a quantified operator. As the function μ is recursive, $\mu(a[\overrightarrow{T}])$ may return 0 in one or more components when $a[\overrightarrow{T}]$ is a sub-PASTD. In that case, it means that the associated parameter T_i is not used in $a[\overrightarrow{T}]$. Thus, it should be possible to find a sub-state of rank $\mu(a[\overrightarrow{T}])$ because the empty parameter value will have no consequence in the construction of the sub-state. For instance, the PASTD (elem) with parameters $(T_1, ..., T_k)$ has maximum transition size $(0, ..., 0)$ and we can find a state $s = (\mathsf{elem}_\circ)$ such that $s.val = (\emptyset, ..., \emptyset)$.

The function μ will allow us to prove the backward-downward monotony. But first, let us examine some properties of the set of states $\mathcal{Q}^\sharp$. Thanks to the extension of the state space, we are able to construct abstract states from a concrete one by either extending the parameter value without changing the state expression or by pruning, in some cases, some branches of a state expression. Those two operations are essential to the proof of the bdm and are represented by the following lemmas.

Lemma 19 (Lifting)**.** *Let $a[\overrightarrow{T}]$ be a PASTD and $\overrightarrow{V}$ a parameter value. Let $q_1, q_2 \in \mathcal{Q}^\sharp$ such that $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and $q_1 \sqsubseteq q_2$. For all value $\overrightarrow{V}$ such that $q_1.val \subseteq \overrightarrow{V} \subseteq q_2.val$, there exists $q \in \mathcal{Q}^\sharp$ such that $q_1 \sqsubseteq q \sqsubseteq q_2$ and $q.val = \overrightarrow{V}$.*

Proof. Take $q \in \mathcal{Q}^\sharp$ such that q has the same state expression and same PASTD as q_1 but with $q.val = \overrightarrow{V}$. This is possible by definition of $\mathcal{Q}^\sharp$, which allows the domain of the function of the quantified interleaving operator to be partial. We still have that $q \sqsubseteq q_2$. □

Lemma 19 allows to easily pick an intermediary state for a given parameter value between two ordered states. It does not hold within $\mathcal{Q}$ as pruning some branches (of the quantified interleaving operator) from the greater state q_2 may results into a partially defined state. On the other hand, it is not always possible to scale a state down to any parameter value, but we can find a smaller state whose rank is bounded by a specific value. Indeed, for all state, there is a smaller one whose size is bounded by the value given by the function of Definition 43.

Lemma 20 (Pruning)**.** *Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{T}]$, there exists $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$ and $\gamma(q) \leq_k c$.*

Remark that the maximum transition size coincides with the upper bound for the smallest valid part of a state, that we will prove further. First, let us show that pruning also preserves final states.

Lemma 21 (Pruning final)**.** *Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{T}]$, if $final(s)$, then there exists $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$, $\gamma(q) \leq_k c$ and $final(q)$.*

Considering the previous lemmas, we are now able to show the bdm for $\mathcal{S}_a^\sharp$. Detailed proofs are given in Appendix B.5.

Lemma 22. *Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s_1, s_2 \in \mathcal{Q}^\sharp$ such that $s_1.pa = s_2.pa = a[\overrightarrow{T}]$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, there exist $q_1, q_2 \in \mathcal{Q}^\sharp$ such that $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and $q_1 \xrightarrow{\sigma,\Gamma} q_2$ and:*

1. *$\gamma(q_1) = \gamma(q_2) \leq_k c$, and*
2. *$q_1 \preceq s_1$ and $q_2 \preceq s_2$, and*
3. *for all $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \preceq q_2' \preceq s_2$, there exists $q_1' \in \mathcal{Q}^\sharp$ such that $q_1 \preceq q_1' \preceq s_1$ and $q_1' \xrightarrow{\sigma',\Gamma'} q_2'$, where σ' and Γ' are the renamed event and environment with regards to the rename function ξ such that $q_2 \sqsubseteq \xi(q_2')$.*

Lemma 23. *Let $a[\overrightarrow{T}]$ be a PASTD, $\tilde{\mathcal{S}}_a = (\mathcal{T}_a/{\sim}, \rightarrow, \preceq)$ and $c = \mu(a[\overrightarrow{T}])$. For each transition $\tilde{s_1} \rightarrow \tilde{s_2}$ in $\tilde{\mathcal{S}}_a$, there is $\tilde{q_1} \rightarrow \tilde{q_2}$ in $\tilde{\mathcal{S}}_a$ such that $\tilde{q_1} \preceq \tilde{s_1}$, $\tilde{q_2} \preceq \tilde{s_2}$, $\gamma(\tilde{q_1}) \leq c$, $\gamma(\tilde{q_2}) \leq c$, and $\forall \tilde{q_2'} \in \mathcal{T}_a/{\sim} \cdot \tilde{q_2} \preceq \tilde{q_2'} \preceq \tilde{s_2} \implies \exists \tilde{q_1'} \in \mathcal{T}_a/{\sim} \cdot \tilde{q_1'} \rightarrow \tilde{q_2'} \wedge \tilde{q_1} \preceq \tilde{q_1'} \preceq \tilde{s_1}$.*

Proof. By application of the env rule and Lemma 22. □

Now, let us prove that Condition 2 of Definition 14 is satisfied, *i.e.* if two PASTD states have an upper bound, they have a "bounded" one according to the rank function γ.

Lemma 24. *Let $a[\overrightarrow{T}]$ be a PASTD and $\tilde{\mathcal{S}}_a = (\mathcal{T}_a/{\sim}, \to, \preceq)$. For all $\tilde{s_1}, \tilde{s_2}, \tilde{s_3} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s_1} \preceq \tilde{s_3}$ and $\tilde{s_2} \preceq \tilde{s_3}$, there exists $\tilde{s_4} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s_1} \preceq \tilde{s_4}$, $\tilde{s_2} \preceq \tilde{s_4}$, $\tilde{s_4} \preceq \tilde{s_3}$ and $\gamma(\tilde{s_4}) \leq \gamma(\tilde{s_1}) + \gamma(\tilde{s_2})$.*

Proof. Let $\tilde{s_1}, \tilde{s_2}, \tilde{s_3} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s_1} \preceq \tilde{s_3}$ and $\tilde{s_2} \preceq \tilde{s_3}$. Then, there is $s_1 \in \tilde{s_1}, s_2 \in \tilde{s_2}, s_3 \in \tilde{s_3}$ such that $s_1 \preceq s_3$ and $s_2 \preceq s_3$. We construct s_4 as the least state $s_4 \sqsubseteq s_3$ such that $s_1 \preceq s_4$ and $s_2 \preceq s_4$. s_4 is a well-formed state because s_1, s_2 and s_3 are from the same PASTD and only branches from quantified interleaving nodes are pruned from s_3 to obtain s_4. Hence, $s_4 \in \mathcal{T}_a$. By definition of γ, we have that $\gamma(s_4) \leq \gamma(s_1) + \gamma(s_2)$. Take $\tilde{s_4} \in \mathcal{T}_a/{\sim}$ the equivalence class of s_4. We have $\tilde{s_1} \preceq \tilde{s_4}$, $\tilde{s_2} \preceq \tilde{s_4}$, $\tilde{s_4} \preceq \tilde{s_3}$ and $\gamma(\tilde{s_4}) \leq \gamma(\tilde{s_1}) + \gamma(\tilde{s_2})$. □

Intuitively, take the example of the library and consider a state s_1 such that $\gamma(s_1) = (1,1)$ and where the member m_1 borrowed the book b_1 and a state s_2 such that $\gamma(s_2) = (1,1)$ and where the member m_2 borrowed the book b_2, then clearly s_1 and s_2 have upper bounds. Besides, we can construct a "little" state s_3 such that $\gamma(s_3) = (2,2)$ including the two different loans.

We can now state that PASTD are effective RMTS.

Theorem 12. *Let $a[\overrightarrow{T}]$ be a PASTD and $\tilde{\mathcal{S}}_a = (\mathcal{T}_a/{\sim}, \to, \preceq)$ the quotient MTS, then $(\mathcal{T}_a/{\sim}, \to, \preceq, \gamma, \mu(a[\overrightarrow{T}]), \overrightarrow{0})$ is an effective RMTS.*

Proof. Let $\mathcal{S} = (\mathcal{T}_a/{\sim}, \to, \preceq, \gamma, \mu(a[\overrightarrow{T}]), \overrightarrow{0})$. $(\mathcal{T}_a/{\sim}, \to, \preceq)$ is a MTS by Proposition 16 and $\preceq$ a po on $\mathcal{T}_a/{\sim}$.

1. Let $\tilde{s}, \tilde{s'} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s} \preceq \tilde{s'}$. We clearly have $\gamma(\tilde{s}) \leq \gamma(\tilde{s'})$ by definition of γ. Thus, γ is monotonic. Let $r \in \mathbb{N}^k$. $\gamma^{-1}(r)$ is finite by a combinatorial argument.

2. By Lemma 24, for all $\tilde{s_1}, \tilde{s_2}, \tilde{s_3} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s_1} \preceq \tilde{s_3}$ and $\tilde{s_2} \preceq \tilde{s_3}$, there exists $\tilde{s_4} \in \mathcal{T}_a/{\sim}$ such that $\tilde{s_1} \preceq \tilde{s_4}$, $\tilde{s_2} \preceq \tilde{s_4}$, $\tilde{s_4} \preceq \tilde{s_3}$ and $\gamma(\tilde{s_4}) \leq \gamma(\tilde{s_1}) + \gamma(\tilde{s_2})$.

3. For all $\tilde{s_1} \to \tilde{s_2}$, $\gamma(\tilde{s_1}) \leq \gamma(\tilde{s_2}) + \overrightarrow{0}$ because $\gamma(\tilde{s_1}) = \gamma(\tilde{s_2})$ by Proposition 17.

4. By Lemma 23, for all $\tilde{s_1} \to \tilde{s_2}$, there is $\tilde{q_1} \to \tilde{q_2}$ such that $\tilde{q_1} \preceq \tilde{s_1}$, $\tilde{q_2} \preceq \tilde{s_2}$, $\gamma(\tilde{q_1}) \leq \mu(a[\overrightarrow{T}])$, $\gamma(\tilde{q_2}) \leq \mu(a[\overrightarrow{T}])$, and $\forall \tilde{q'_2} \in \mathcal{T}_a/{\sim} \cdot \tilde{q_2} \preceq \tilde{q'_2} \preceq \tilde{s_2} \implies \exists \tilde{q'_1} \in \mathcal{T}_a/{\sim} \cdot \tilde{q'_1} \to \tilde{q'_2} \land \tilde{q_1} \preceq \tilde{q'_1} \preceq \tilde{s_1}$.

As a consequence, $\mathcal{S}$ is a RMTS. The transition relation is decidable as it is defined by a set of rules in [23]. Also by definition, $\preceq$ is decidable and $\gamma(\tilde{s})$ computable for $\tilde{s} \in \mathcal{T}_a/{\sim}$. For $r \in \mathbb{N}^k$, $\gamma^{-1}(r)$ is computable by enumerating all well-formed states $\tilde{s}$ such that $\gamma(\tilde{s}) = r$. Thus, $\mathcal{S}$ is an effective RMTS. □

For any PASTD, we can conclude that we can compute a finite pred-basis of any state s. However, if we want to compute a finite basis for $Pred^*(\uparrow s)$ and decide coverability, we need the wqo condition.

Appendix B

Proofs

B.1 Proofs of Section A.3

Lemma 10. Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter value $\overrightarrow{V}$, $init(a[\overrightarrow{V}]) \in \mathcal{Q}$.

Proof. By structural induction on $a[\overrightarrow{T}]$, we show that each case match Definition 30. Note that $init(a[\overrightarrow{V}]).pa = a[\overrightarrow{T}]$ and $init(a[\overrightarrow{V}]).val = \overrightarrow{V}$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. We have $init(a[\overrightarrow{V}]) = (\mathsf{elem}_\circ) \in \mathcal{Q}$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. By definition of $init$, we have $init(a[\overrightarrow{V}]) = (\mathsf{aut}_\circ, n_0, init(\nu(n_0)[\overrightarrow{V}]))$. By induction hypothesis, we have $init(\nu(n_0)[\overrightarrow{V}]) \in \mathcal{Q}$. Thus, $init(a[\overrightarrow{V}]) \in \mathcal{Q}$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. We have $init(a[\overrightarrow{V}]) = (\star_\circ, \text{false}, \bot) \in \mathcal{Q}$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. We have $init(a[\overrightarrow{V}]) = (|_\circ, \bot, \bot) \in \mathcal{Q}$.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $init(a[\overrightarrow{V}]) = (||_\circ, init(l[\overrightarrow{V}]), init(r[\overrightarrow{V}]))$. By induction hypothesis, $init(l[\overrightarrow{V}]) \in \mathcal{Q}$ and $init(r[\overrightarrow{V}]) \in \mathcal{Q}$. Thus, $init(a[\overrightarrow{V}]) \in \mathcal{Q}$.

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. We have $init(a[\overrightarrow{V}]) = (|:_\circ, \bot, \bot)$. Thus, $init(a[\overrightarrow{V}]) \in \mathcal{Q}$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By cases on X.

   - If $X = T_i \in \overrightarrow{T}$. We have $init(a[\overrightarrow{V}]) = (|||:_\circ, V_i \times \{init(b[\overrightarrow{V}])\})$. By induction hypothesis, $init(b[\overrightarrow{V}]) \in \mathcal{Q}$. Thus, $init(a[\overrightarrow{V}]) \in \mathcal{Q}$.
   - If $X = W$ is not a parameter, we can conclude as well by induction hypothesis.

□

Lemma 11. Let $a[\overrightarrow{T}]$ be a PASTD and $\overrightarrow{V}$ a parameter value. Let s_1, s_2 two states such that $s_1.pa = s_2.pa = a$ and $s_1.val = s_2.val = \overrightarrow{V}$. If $s_1 \in \mathcal{Q}$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then $s_2 \in \mathcal{Q}$.

Proof. Let s_1, s_2 two states, σ an event and Γ an environment such that $s_1 \xrightarrow{\sigma,\Gamma} s_2$ and $s_1 \in \mathcal{Q}$. By structural induction on $s_1.pa = s_2.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. There is no transition from $(\mathsf{elem}_\circ)$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. We have $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_1: $s_2 = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[s_1.val]))$. By Lemma 10, we have $init(\nu(n_2)[s_1.val]) \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case aut_2: $s_2 = (\mathsf{aut}_\circ, n_1, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. By induction hypothesis and as $ss_1 \in \mathcal{Q}$, we have $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_1$: $s_1 = (\star_\circ, started?, ss_1)$, $s_2 = (\star_\circ, \mathrm{true}, ss_2)$ and $init(b[s_1.val]) \xrightarrow{\sigma,\Gamma} ss_2$. By Lemma 10, $init(b[s_1.val]) \in \mathcal{Q}$, and by induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case $\star_2$: $s_1 = (\star_\circ, \mathrm{true}, ss_1)$ and $s_2 = (\star_\circ, \mathrm{true}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. As $ss_1 \in \mathcal{Q}$ and by induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|_1$: $s_1 = (|_\circ, \bot, \bot)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $init(l[s_1.val]) \xrightarrow{\sigma,\Gamma} ss_2$. By Lemma 10, $init(l[s_1.val]) \in \mathcal{Q}$. By induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case $|_2$: same as previous.
   - Case $|_3$: $s_1 = (|_\circ, \mathsf{left}, ss_1)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. We have $ss_1 \in \mathcal{Q}$. By induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case $|_4$: same as previous.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $||_1$: $s_1 = (||_\circ, s_{l1}, s_{r1})$ and $s_2 = (||_\circ, s_{l2}, s_{r1})$ with $\alpha(\sigma) \notin \Delta$ and $s_{l1} \xrightarrow{\sigma,\Gamma} s_{l2}$. We have $s_{l1} \in \mathcal{Q}$ and $s_{r1} \in \mathcal{Q}$. By induction hypothesis, $s_{l2} \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case $||_2$: same.
   - Case $||_3$: similar proof. This time $\alpha(\sigma) \in \Delta$ and the induction hypothesis is used twice (for l and r).

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|:_1$: $s_1 = (|:_\circ, \bot, \bot)$ and $s_2 = (|:_\circ, v, ss_2)$ with $init(b[s_1.val]) \xrightarrow{\sigma,(\![x:=v]\!)\triangleleft\Gamma} ss_2$ and $v \in X$. By Lemma 10, $init(b[s_1.val]) \in \mathcal{Q}$. By induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.
   - Case $|:_2$: $s_1 = (|:_\circ, v, ss_1)$ and $s_2 = (|:_\circ, v, ss_2)$ with $ss_1 \xrightarrow{\sigma,(\![x:=v]\!)\triangleleft\Gamma} ss_2$ and $v \neq \bot$. We have $ss_1 \in \mathcal{Q}$. By induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:_{:}, n, x, X, b)[\overrightarrow{T}]$. By $|||:_1$, the only rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$, $s_1 = (|||:_\circ, f)$ and $s_2 = (|||:_\circ, f \lessdot \{v \mapsto ss_2\})$ with $f(v) \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss_2$. We have $f(v) \in \mathcal{Q}$. By induction hypothesis, $ss_2 \in \mathcal{Q}$. Thus, $s_2 \in \mathcal{Q}$.

□

B.2 Proofs of Section A.4.1

Lemma 12. Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter values $\overrightarrow{V}$ and $\overrightarrow{V'}$ such that $\overrightarrow{V} \subseteq \overrightarrow{V'}$, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

Proof. Let $a[\overrightarrow{T}]$ be a PASTD, $\overrightarrow{V}$ and $\overrightarrow{V'}$ such that $\overrightarrow{V} \subseteq \overrightarrow{V'}$. By structural induction on $a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. We have $\mathit{init}(a[\overrightarrow{V}]) = (\mathsf{elem}_\circ)$ and $\mathit{init}(a[\overrightarrow{V'}]) = (\mathsf{elem}_\circ)$. Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, \mathit{name}, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. By definition, $\mathit{init}(a[\overrightarrow{V}]) = (\mathsf{aut}_\circ, n_0, \mathit{init}(\nu(n_0)[\overrightarrow{V}]))$, $\mathit{init}(a[\overrightarrow{V'}]) = (\mathsf{aut}_\circ, n_0, \mathit{init}(\nu(n_0)[\overrightarrow{V'}]))$. By induction hypothesis, $\mathit{init}(\nu(n_0)[\overrightarrow{V}]) \sqsubseteq \mathit{init}(\nu(n_0)[\overrightarrow{V'}])$. Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$, by definition of $\sqsubseteq$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (\star_\circ, \text{false}, \bot)$. Likewise, $\mathit{init}(a[\overrightarrow{V'}]) = (\star_\circ, \text{false}, \bot)$. Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|_\circ, \bot, \bot)$ and $\mathit{init}(a[\overrightarrow{V'}]) = (|_\circ, \bot, \bot)$.Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (||_\circ, \mathit{init}(l[\overrightarrow{V}]), \mathit{init}(r[\overrightarrow{V}]))$ and $\mathit{init}(a[\overrightarrow{V'}]) = (||_\circ, \mathit{init}(l[\overrightarrow{V'}]), \mathit{init}(r[\overrightarrow{V'}]))$. By induction hypothesis, $\mathit{init}(l[\overrightarrow{V}]) \sqsubseteq \mathit{init}(l[\overrightarrow{V'}])$ and $\mathit{init}(r[\overrightarrow{V}]) \sqsubseteq \mathit{init}(r[\overrightarrow{V'}])$. Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|:_\circ, \bot, \bot)$. Likewise, $\mathit{init}(a[\overrightarrow{V'}]) = (|:_\circ, \bot, \bot)$. Thus, $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By cases on X.
   - If $X = T_i \in \overrightarrow{T}$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|||:_\circ, V_i \times \{\mathit{init}(b[\overrightarrow{V}])\})$ and $\mathit{init}(a[\overrightarrow{V'}]) = (|||:_\circ, V_i' \times \{\mathit{init}(b[\overrightarrow{V'}])\})$. By induction hypothesis, we have $\mathit{init}(b[\overrightarrow{V}]) \sqsubseteq \mathit{init}(b[\overrightarrow{V'}])$. As $V_i \subseteq V_i'$, we conclude that $\mathit{init}(a[\overrightarrow{V}]) \sqsubseteq \mathit{init}(a[\overrightarrow{V'}])$.
   - If $X = W$ is not a parameter, we can conclude as well by induction hypothesis.

□

Lemma 13. For any IPASTD states $s, s' \in \mathcal{Q}$ such that $s \sqsubseteq s'$, if $\mathit{final}(s)$ then $\mathit{final}(s')$.

Proof. Let s, s' such that $s \sqsubseteq s'$. By structural induction on $s.pa$.

1. Base case: $s.pa = (\mathsf{elem})$. We have $s'.pa = (\mathsf{elem})$ and $s' = (\mathsf{elem}_\circ)$. Thus $\mathit{final}(s')$.

2. Inductive case: $s.pa = (\mathsf{aut}, \mathit{name}, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$.
We have $s = (\mathsf{aut}_\circ, n, ss)$, $s' = (\mathsf{aut}_\circ, n, ss')$ with $ss \sqsubseteq ss'$. Assume $\mathit{final}(s)$, *i.e.* by definition $(n \in DF \wedge \mathit{final}(ss)) \vee n \in SF$. By induction hypothesis, if $\mathit{final}(ss)$ then $\mathit{final}(ss')$. We can conclude that $(n \in DF \wedge \mathit{final}(ss')) \vee n \in SF$, *i.e.* $\mathit{final}(s')$ is true.

3. Inductive case: $s.pa = (\star, n, b)[\overrightarrow{T}]$. We have $s = (\star_\circ, \mathit{started?}, ss)$ and $s' = (\star_\circ, \mathit{started?}, ss')$. Assume $\mathit{final}(s)$, *i.e.* $\mathit{final}(ss) \vee \neg \mathit{started?}$. By induction hypothesis, we have $\mathit{final}(ss')$, as $ss \sqsubseteq ss'$. Thus, $\mathit{final}(s')$.

4. Inductive case: $s.pa = (|, n, l, r)[\overrightarrow{T}]$. By cases on s:
   - If $s = (|_\circ, \bot, \bot)$, then $s' = (|_\circ, \bot, \bot)$. Suppose that $\mathit{final}(s)$ is true, that is $\mathit{final}(\mathit{init}(l[s.\mathit{val}])) \vee \mathit{final}(\mathit{init}(r[s.\mathit{val}]))$.
     - Case $\mathit{final}(\mathit{init}(l[s.\mathit{val}]))$. By Lemma 12, $\mathit{init}(l[s.\mathit{val}]) \sqsubseteq \mathit{init}(l[s'.\mathit{val}])$, because $s.\mathit{val} \subseteq s'.\mathit{val}$. So, by induction hypothesis, $\mathit{final}(\mathit{init}(l[s'.\mathit{val}]))$ is true. Thus, $\mathit{final}(s')$ is true by definition.
     - Same reasoning for $\mathit{final}(\mathit{init}(r[s.\mathit{val}]))$.
   - If $s = (|_\circ, \mathsf{left}, ss)$, then $s' = (|_\circ, \mathsf{left}, ss')$ with $ss \sqsubseteq ss'$. Assume $\mathit{final}(s)$, *i.e.* $\mathit{final}(ss)$. By induction hypothesis, $\mathit{final}(ss')$ is true, and is equivalent to $\mathit{final}(s')$ by definition.
   - Same for $s = (|_\circ, \mathsf{right}, ss)$.

5. Inductive case: $s.pa = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $s = (||_\circ, s_l, s_r)$ and $s' = (||_\circ, s'_l, s'_r)$ with $s_l \sqsubseteq s'_l$ and $s_r \sqsubseteq s'_r$. Assume $\mathit{final}(s)$, *i.e.* $\mathit{final}(s_l) \wedge \mathit{final}(s_r)$. By induction hypothesis, $\mathit{final}(s'_l)$ and $\mathit{final}(s'_r)$. Thus, $\mathit{final}(s')$.

6. Inductive case: $s.pa = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on s:
   - If $s = (|:_\circ, \bot, \bot)$, then $s' = (|:_\circ, \bot, \bot)$.
   Assume $\mathit{final}(s)$, *i.e.* $\mathit{final}(\mathit{init}(b[s.\mathit{val}]))$. By Lemma 12 and induction hypothesis, $\mathit{final}(\mathit{init}(b[s'.\mathit{val}]))$. Thus, $\mathit{final}(s')$.
   - If $s = (|:_\circ, v, ss)$, $s' = (|:_\circ, v, ss')$ with $ss \sqsubseteq ss'$. Assume $\mathit{final}(s)$, *i.e.* $\mathit{final}(ss)$. By induction hypothesis, $\mathit{final}(ss')$, *i.e.* $\mathit{final}(s')$.

7. Inductive case: $s.pa = (|||:, n, x, X, b)[\overrightarrow{T}]$. We have $s = (|||:_\circ, f)$ and $s' = (|||:_\circ, f')$ with $\mathit{dom}(f) \subseteq \mathit{dom}(f')$ and for all $v \in \mathit{dom}(f)$, $f(v) \sqsubseteq f'(v)$. Suppose $\mathit{final}(s)$. By Definition 34 and by cases on X:
   - If X is a parameter, then there exists $v \in \mathit{dom}(f)$ such that $\mathit{final}(f(v))$. By induction hypothesis, $\mathit{final}(f'(v))$. Thus, $\mathit{final}(s')$.
   - Else, $\mathit{dom}(f) = \mathit{dom}(f')$ and for all $v \in \mathit{dom}(f)$ we have $\mathit{final}(f(v))$. By induction hypothesis, for all $v \in \mathit{dom}(f)$, $\mathit{final}(f'(v))$. Thus, $\mathit{final}(s')$.

□

Lemma 14. For any IPASTD states $s_1, s'_1, s_2 \in \mathcal{Q}$, any event σ and any environment Γ, if $s_1 \sqsubseteq s'_1$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then there exists s'_2 such that $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.

Proof. Let s_1, s_2, s'_1 three states, σ an event and Γ an environment such that $s_1 \sqsubseteq s'_1$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$. By structural induction on $s_1.pa = s_2.pa = s'_1.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. There is no transition from $(\mathsf{elem}_\circ)$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. We have $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$, $s'_1 = (\mathsf{aut}_\circ, n_1, ss'_1)$ with $ss_1 \sqsubseteq ss'_1$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_1: $s_2 = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[s_1.val]))$ with $\delta(n_1, n_2, \sigma', final?)$ and $final? \Rightarrow final(ss_1)$ and $\sigma'[\Gamma] = \sigma$. By Lemma 13, $final(ss_1) \Rightarrow final(ss'_1)$. And, by the same inference rule, we have $(\mathsf{aut}_\circ, n_1, ss'_1) \xrightarrow{\sigma,\Gamma} s'_2$ such that $s'_2 = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[s'_1.val]))$. Thus, $s_2 \sqsubseteq s'_2$ by Lemma 12 and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.
   - Case aut_2: $s_2 = (\mathsf{aut}_\circ, n_1, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. By induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $ss'_1 \xrightarrow{\sigma,\Gamma} ss'_2$. Let $s'_2 = (\mathsf{aut}_\circ, n_1, ss'_2)$. We have $s_2 \sqsubseteq s'_2$ by definition of $\sqsubseteq$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$ by the rule aut_2.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_1$: $s_1 = (\star_\circ, started?, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $final(ss_1) \vee \neg started?$ and $init(b[s_1.val]) \xrightarrow{\sigma,\Gamma} ss_2$. We have $s'_1 = (\star_\circ, started?, ss'_1)$ with $ss_1 \sqsubseteq ss'_1$. If $final(ss_1)$ then $final(ss'_1)$ by Lemma 13. Thus, $final(ss'_1) \vee \neg started?$. By Lemma 12 and induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $init(b[s'_1.val]) \xrightarrow{\sigma,\Gamma} ss'_2$. Let $s'_2 = (\star_\circ, \text{true}, ss'_2)$. We have $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$ by the rule $\star_1$.
   - Case $\star_2$: $s_1 = (\star_\circ, \text{true}, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. We have $s'_1 = (\star_\circ, \text{true}, ss'_1)$ with $ss_1 \sqsubseteq ss'_1$. By induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $ss'_1 \xrightarrow{\sigma,\Gamma} ss'_2$. Let $s'_2 = (\star_\circ, \text{true}, ss'_2)$. We have $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|_1$: $s_1 = (|_\circ, \bot, \bot)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $init(l[s_1.val]) \xrightarrow{\sigma,\Gamma} ss_2$. We have $s'_1 = (|_\circ, \bot, \bot)$. And by Lemma 12, $init(l[s_1.val]) \sqsubseteq init(l[s'_1.val])$ because $s_1 \sqsubseteq s'_1$. Thus, by induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $init(l[s'_1.val]) \xrightarrow{\sigma,\Gamma} ss'_2$. Let $s'_2 = (|_\circ, \mathsf{left}, ss'_2)$. We can conclude that $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.
   - Case $|_2$: same as previous.
   - Case $|_3$: $s_1 = (|_\circ, \mathsf{left}, ss_1)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. We have $s'_1 = (|_\circ, \mathsf{left}, ss'_1)$ with $ss_1 \sqsubseteq ss'_1$. By induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $ss'_1 \xrightarrow{\sigma,\Gamma} ss'_2$. Let $s'_2 = (|_\circ, \mathsf{left}, ss'_2)$. Thus, $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.
   - Case $|_4$: same as previous.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $||_1$: $s_1 = (||_\circ, s_{l1}, s_{r1})$ and $s_2 = (||_\circ, s_{l2}, s_{r1})$ with $\alpha(\sigma) \notin \Delta$ and $s_{l1} \xrightarrow{\sigma,\Gamma} s_{l2}$. We have $s'_1 = (||_\circ, s_{l}{}'_1, s_{r}{}'_1)$ with $s_{l1} \sqsubseteq s_{l}{}'_1$ and $s_{r1} \sqsubseteq s_{r}{}'_1$. By induction hypothesis, there exists $s_{l}{}'_2$ such that $s_{l2} \sqsubseteq s_{l}{}'_2$ and $s_{l}{}'_1 \xrightarrow{\sigma,\Gamma} s_{l}{}'_2$. Let $s'_2 = (||_\circ, s_{l}{}'_2, s_{r}{}'_1)$. We have $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$ by the rule $||_1$ and $s_2 \sqsubseteq s'_2$.
   - Case $||_2$: same.
   - Case $||_3$: similar proof. This time $\alpha(\sigma) \in \Delta$ and the induction hypothesis is used twice (for l and r).

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|:_1$: $s_1 = (|:_\circ, \bot, \bot)$ and $s_2 = (|:_\circ, v, ss_2)$ with $init(b[s_1.val]) \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss_2$ and $v \in X$. We have $s'_1 = (|:_\circ, \bot, \bot)$ with $s_1.val \subseteq s'_1.val$. And by Lemma 12, $init(b[s_1.val]) \sqsubseteq init(b[s'_1.val])$. Then, by induction hypothesis, there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $init(b[s'_1.val]) \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss'_2$. Let $s'_2 = (|:_\circ, v, ss'_2)$. We have $s_2 \sqsubseteq s'_2$. By cases on X:
     - If $X = V_i \in s_1.val$, then $v \in V'_i \supseteq V_i$ with $V_i \in s_1.val$ and $V'_i \in s'_1.val$. Thus by the rule $|:_1$, $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.
     - If $X = W \notin s_1.val$, then $v \in W$ holds and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.
   - Case $|:_2$: $s_1 = (|:_\circ, v, ss_1)$ and $s_2 = (|:_\circ, v, ss_2)$ with $ss_1 \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss_2$ and $v \neq \bot$. We have $s'_1 = (|:_\circ, v, ss'_1)$ with $ss_1 \sqsubseteq ss'_1$. By induction hypothesis, there is ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $ss'_1 \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss'_2$. Let $s'_2 = (|:_\circ, v, ss'_2)$. We have $s_2 \sqsubseteq s'_2$ and $s'_1 \xrightarrow{\sigma,\Gamma} s'_2$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By $|||:_1$, the only rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$, $s_1 = (|||:_\circ, f)$ and $s_2 = (|||:_\circ, f \Lleftarrow \{v \mapsto ss_2\})$ with $f(v) \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss_2$. We have $s'_1 = (|||:_\circ, f')$ with $dom(f) \subseteq dom(f')$ and for all $w \in dom(f)$, $f(w) \sqsubseteq f'(w)$. As $f(v) \sqsubseteq f'(v)$, we use induction hypothesis to conclude that there exists ss'_2 such that $ss_2 \sqsubseteq ss'_2$ and $f'(v) \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss'_2$. By the rule $|||:_1$, we have $(|||:_\circ, f') \xrightarrow{\sigma,\Gamma} (|||:_\circ, f' \Lleftarrow \{v \mapsto ss'_2\})$. And by definition of $\sqsubseteq$, $(|||:_\circ, f \Lleftarrow \{v \mapsto ss_2\}) \sqsubseteq (|||:_\circ, f' \Lleftarrow \{v \mapsto ss'_2\})$.

□

B.3 Proofs of Section A.4.2

Lemma 15. Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter value $\overrightarrow{V}$ and any rename function ξ defined by permutations $\rho_1, ..., \rho_k$, we have

$$\xi(init(a[\overrightarrow{V}])) = init(a[\rho_1(V_1), ..., \rho_k(V_k)])$$

Proof. Let $a[\overrightarrow{T}]$ be a PASTD, $\overrightarrow{V}$ a parameter value and ξ a rename function given by permutations $\rho_1, ..., \rho_k$. For $\overrightarrow{V} = (V_1, ..., V_k)$, let $\rho(\overrightarrow{V}) = (\rho_1(V_1), ..., \rho_k(V_k))$. By structural induction on $a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. We have $\mathit{init}(a[\overrightarrow{V}]) = (\mathsf{elem}_\circ)$. Thus, $\xi(\mathit{init}(a[\overrightarrow{V}])) = (\mathsf{elem}_\circ)$ such that $\xi(\mathit{init}(a[\overrightarrow{V}])).\mathit{val} = \rho(\overrightarrow{V})$ and $\mathit{init}(a[\rho(\overrightarrow{V})]) = (\mathsf{elem}_\circ)$ such that $\mathit{init}(a[\rho(\overrightarrow{V})]).\mathit{val} = \rho(\overrightarrow{V})$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, \mathit{name}, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$.
   By definition, w have $\mathit{init}(a[\overrightarrow{V}]) = (\mathsf{aut}_\circ, n_0, \mathit{init}(\nu(n_0)[\overrightarrow{V}]))$ and
   $\xi(\mathit{init}(a[\overrightarrow{V}])) = (\mathsf{aut}_\circ, n_0, \xi(\mathit{init}(\nu(n_0)[\overrightarrow{V}])))$. Moreover,
   $\mathit{init}(a[\rho(\overrightarrow{V})]) = (\mathsf{aut}_\circ, n_0, \mathit{init}(\nu(n_0)[\rho(\overrightarrow{V})]))$. By the induction hypothesis,
   $(\mathsf{aut}_\circ, n_0, \xi(\mathit{init}(\nu(n_0)[\overrightarrow{V}]))) = (\mathsf{aut}_\circ, n_0, \mathit{init}(\nu(n_0)[\rho(\overrightarrow{V})]))$. Finally, we have
   $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (\star_\circ, \text{false}, \bot)$. Moreover, $\xi(\mathit{init}(a[\overrightarrow{V}])) = (\star_\circ, \text{false}, \bot)$ with $\xi(\mathit{init}(a[\overrightarrow{V}])).\mathit{val} = \rho(\overrightarrow{V})$. Thus,
   $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|_\circ, \bot, \bot)$. Moreover, $\xi(\mathit{init}(a[\overrightarrow{V}])) = (|_\circ, \bot, \bot)$ with $\xi(\mathit{init}(a[\overrightarrow{V}])).\mathit{val} = \rho(\overrightarrow{V})$. Thus, $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (||_\circ, \mathit{init}(l[\overrightarrow{V}]), \mathit{init}(r[\overrightarrow{V}]))$ and $\xi(\mathit{init}(a[\overrightarrow{V}])) = (||_\circ, \xi(\mathit{init}(l[\overrightarrow{V}])), \xi(\mathit{init}(r[\overrightarrow{V}])))$. By induction hypothesis, we have that $\xi(\mathit{init}(a[\overrightarrow{V}])) = (||_\circ, \mathit{init}(l[\rho(\overrightarrow{V})]), \mathit{init}(r[\rho(\overrightarrow{V})]))$. And as $\mathit{init}(a[\rho(\overrightarrow{V})]) = (||_\circ, \mathit{init}(l[\rho(\overrightarrow{V})]), \mathit{init}(r[\rho(\overrightarrow{V})]))$, then $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|:_\circ, \bot, \bot)$. Moreover, $\xi(\mathit{init}(a[\overrightarrow{V}])) = (|:_\circ, \bot, \bot)$ with $\xi(\mathit{init}(a[\overrightarrow{V}])).\mathit{val} = \rho(\overrightarrow{V})$. Thus, $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By cases on X.

   - If $X = T_i \in \overrightarrow{T}$. We have $\mathit{init}(a[\overrightarrow{V}]) = (|||:_\circ, V_i \times \{\mathit{init}(b[\overrightarrow{V}])\})$ and $\xi(\mathit{init}(a[\overrightarrow{V}])) = (|||:_\circ, \rho_i(V_i) \times \{\xi(\mathit{init}(b[\overrightarrow{V}]))\})$. By induction hypothesis, we have $\xi(\mathit{init}(b[\overrightarrow{V}])) = \mathit{init}(b[\rho(\overrightarrow{V})])$. Thus, $\xi(\mathit{init}(a[\overrightarrow{V}])) = \mathit{init}(a[\rho(\overrightarrow{V})])$.
   - If $X = W$ is not a parameter, we can conclude as well by induction hypothesis.

□

Lemma 16. For any $s \in \mathcal{Q}$ and for any rename function ξ, if $\mathit{final}(s)$, then $\mathit{final}(\xi(s))$.

Proof. Let s be an IPASTD state and ξ a rename function given by permutations $\rho_1, ..., \rho_k$. For a parameter value $\overrightarrow{V} = (V_1, ..., V_k)$, let $\rho(\overrightarrow{V}) = (\rho_1(V_1), ..., \rho_k(V_k))$. By structural induction on $s.pa$.

1. Base case: $s.pa = (\mathsf{elem})$. We have $\xi(s) = (\mathsf{elem}_\circ)$ and $\mathit{final}(\xi(s))$.

2. Inductive case: $s.pa = (\mathsf{aut}, \mathit{name}, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$.
   We have $s = (\mathsf{aut}_\circ, n, ss)$. Assume $\mathit{final}(s)$, *i.e.* $(n \in DF \wedge \mathit{final}(ss)) \vee n \in SF$. By induction hypothesis, $\mathit{final}(\xi(ss))$. We have $\xi(s) = (\mathsf{aut}_\circ, n, \xi(ss))$. Thus, we conclude $\mathit{final}(\xi(s))$ by definition of final.

3. Inductive case: $s.pa = (\star, n, b)[\overrightarrow{T}]$. We have $s = (\star_\circ, started?, ss)$ and $\xi(s) = (\star_\circ, started?, \xi(ss))$. Assume $final(s)$, *i.e.* $final(ss) \lor \neg started?$. By induction hypothesis, we have $final(\xi(ss))$. Thus, $final(\xi(s))$.

4. Inductive case: $s.pa = (|, n, l, r)[\overrightarrow{T}]$. By cases on s:

   - If $s = (|_\circ, \bot, \bot)$, then $\xi(s) = (|_\circ, \bot, \bot)$ with $\xi(s).val = \rho(s.val)$. Suppose $final(s)$, *i.e.* $final(init(l[s.val])) \lor final(init(r[s.val]))$.
     - Case $final(init(l[s.val]))$.
       By induction hypothesis, $final(\xi(init(l[s.val])))$. So by Lemma 15, $final(init(l[\rho(s.val)]))$. Thus, $final(\xi(s))$ is true by definition.
     - Same reasoning for $final(init(r[s.val]))$.
   - If $s = (|_\circ, \mathsf{left}, ss)$, then $\xi(s) = (|_\circ, \mathsf{left}, \xi(ss))$. Assume $final(s)$, *i.e.* $final(ss)$. By induction hypothesis, $final(\xi(ss))$ is true, and is equivalent to $final(\xi(s))$ by definition.
   - Same for $s = (|_\circ, \mathsf{right}, ss)$.

5. Inductive case: $s.pa = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $s = (||_\circ, s_l, s_r)$ and $\xi(s) = (||_\circ, \xi(s_l), \xi(s_r))$. Assume $final(s)$, *i.e.* $final(s_l) \land final(s_r)$. By induction hypothesis, $final(\xi(s_l))$ and $final(\xi(s_r))$. Thus, $final(\xi(s))$.

6. Inductive case: $s.pa = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on s:

   - If $s = (|:_\circ, \bot, \bot)$, then $\xi(s) = (|:_\circ, \bot, \bot)$ with $\xi(s).val = \rho(s.val)$. Assume $final(s)$, *i.e.* $final(init(b[s.val]))$. Thus, $final(init(b[s.val]))$. By Lemma 15 and induction hypothesis, we conclude $final(init(b[\rho(s.val)]))$.
     Thus, $final(\xi(s))$.
   - If $s = (|:_\circ, v, ss)$, then $\xi(s) = (|:_\circ, \_, \xi(ss))$. Assume $final(s)$, *i.e.* $final(ss)$. By induction hypothesis, $final(\xi(ss))$, *i.e.* $final(\xi(s))$.

7. Inductive case: $s.pa = (|||:, n, x, X, b)[\overrightarrow{T}]$. We have $s = (|||:_\circ, f)$ and $\xi(s) = (|||:_\circ, f')$. By cases on X:

   - If $X = T_i$ is a parameter, then $dom(f') = \rho_i(dom(f))$ and for all $v \in dom(f)$ we have $f'(\rho_i(v)) = \xi(f(v))$. Suppose $final(s)$. By Definition 34, there exists $v \in dom(f)$ such that $final(f(v))$. By induction hypothesis, $final(f'(\rho_i(v)))$. Thus, $final(\xi(s))$.
   - Else, $dom(f) = dom(f')$ and for all $v \in dom(f)$ we have $f'(v) = \xi(f(v))$. Suppose $final(s)$. By Definition 34, for all $v \in dom(f)$ we have $final(f(v))$. By induction hypothesis, $final(f'(v))$. Thus, $final(\xi(s))$.

□

Lemma 17. Let $P_1, ..., P_k$ a set of parameter domains. For any states $s_1, s_2 \in \mathcal{Q}$, any rename function ξ defined by permutations $\rho_1, ..., \rho_k$ on $P_1, ..., P_k$ respectively, any event σ and any environment $\Gamma = (\!|x_1, ..., x_n := v_1, ..., v_n|\!)$, if $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$, where $\sigma' = \sigma[v_1, ..., v_n := v'_1, ..., v'_n]$ and $\Gamma' = (\!|x_1, ..., x_n := v'_1, ..., v'_n|\!)$ with $v'_i = \rho_j(v_i)$ if $\exists j \cdot v_i \in P_j$ or $v'_i = v_i$ else.

Proof. Let $P_1, ..., P_k$ a set of parameter domains. Let $s_1, s_2 \in \mathcal{Q}$, ξ defined by permutations $\rho_1, ..., \rho_k$ on $P_1, ..., P_k$, σ an event and $\Gamma = (\![x_1, ..., x_n := v_1, ..., v_n]\!)$ an environment, such that $s_1 \xrightarrow{\sigma,\Gamma} s_2$. Let $\sigma' = \sigma[v_1, ..., v_n := v'_1, ..., v'_n]$ and $\Gamma' = (\![x_1, ..., x_n := v'_1, ..., v'_n]\!)$ with $v'_i = \rho_j(v_i)$ if $\exists j \cdot v_i \in P_j$ or $v'_i = v_i$ else. Let $\overrightarrow{V} = s_1.val = s_2.val$ and $\overrightarrow{V'} = \rho(\overrightarrow{V})$. By structural induction on $s_1.pa = s_2.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. There is no transition from $(\mathsf{elem}_\circ)$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$, $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $\xi(s_1) = (\mathsf{aut}_\circ, n_1, \xi(ss_1))$ with $\xi(s_1).val = \overrightarrow{V'}$. By cases on inference rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_1: $s_2 = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[\overrightarrow{V}]))$ with $\delta(n_1, n_2, \sigma'', final?)$ and $final? \Rightarrow final(ss_1)$ and $\sigma''[\Gamma] = \sigma$.
     We have $\xi(s_2) = (\mathsf{aut}_\circ, n_2, \xi(init(\nu(n_2)[\overrightarrow{V}])))$ and
     $\xi(s_2) = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[\overrightarrow{V'}]))$ by Lemma 15, with $\xi(s_2).val = \overrightarrow{V'}$. Besides, by Lemma 16, $final(ss_1) \Rightarrow final(\xi(ss_1))$. Moreover, $\sigma''[\Gamma'] = \sigma[v_1, ..., v_n := v'_1, ..., v'_n] = \sigma'$. Thus, by the same inference rule, we have $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
   - Case aut_2: $s_2 = (\mathsf{aut}_\circ, n_1, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. By induction hypothesis, we have $\xi(ss_1) \xrightarrow{\sigma',\Gamma'} \xi(ss_2)$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$ by the rule aut_2.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_1$: $s_1 = (\star_\circ, started?, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $final(ss_1) \vee \neg started?$ and $init(b[\overrightarrow{V}]) \xrightarrow{\sigma,\Gamma} ss_2$. We have $\xi(s_1) = (\star_\circ, started?, \xi(ss_1))$. If $final(ss_1)$ then $final(\xi(ss_1))$ by Lemma 16. Thus, $final(\xi(ss_1)) \vee \neg started?$. By Lemma 15 and induction hypothesis, $init(b[\overrightarrow{V'}]) \xrightarrow{\sigma',\Gamma'} \xi(ss_2)$. We have $\xi(s_2) = (\star_\circ, \text{true}, \xi(ss_2))$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$ by the rule $\star_1$.
   - Case $\star_2$: $s_1 = (\star_\circ, \text{true}, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. We have $\xi(s_1) = (\star_\circ, \text{true}, \xi(ss_1))$. By induction hypothesis, $\xi(ss_1) \xrightarrow{\sigma',\Gamma'} \xi(ss_2)$. We have $\xi(s_2) = (\star_\circ, \text{true}, \xi(ss_2))$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|_1$: $s_1 = (|_\circ, \bot, \bot)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $init(l[\overrightarrow{V}]) \xrightarrow{\sigma,\Gamma} ss_2$. We have $\xi(s_1) = (|_\circ, \bot, \bot)$. By Lemma 15, $\xi(init(l[\overrightarrow{V}])) = init(l[\overrightarrow{V'}])$. Thus, by induction hypothesis, $init(l[\overrightarrow{V'}]) \xrightarrow{\sigma',\Gamma'} \xi(ss_2)$. We have $\xi(s_2) = (|_\circ, \mathsf{left}, \xi(ss_2))$. We can conclude that $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
   - Case $|_2$: same as previous.
   - Case $|_3$: $s_1 = (|_\circ, \mathsf{left}, ss_1)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. We have $\xi(s_1) = (|_\circ, \mathsf{left}, \xi(ss_1))$ and $\xi(s_2) = (|_\circ, \mathsf{left}, \xi(ss_2))$. By induction hypothesis, $\xi(ss_1) \xrightarrow{\sigma',\Gamma'} \xi(ss_2)$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.

- Case $|_4$: same as previous.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $||_1$: $s_1 = (||_\circ, s_{l1}, s_{r1})$ and $s_2 = (||_\circ, s_{l2}, s_{r1})$ with $\alpha(\sigma) \notin \Delta$ and $s_{l1} \xrightarrow{\sigma,\Gamma} s_{l2}$. We have $\xi(s_1) = (||_\circ, \xi(s_{l1}), \xi(s_{r1}))$ and $\xi(s_2) = (||_\circ, \xi(s_{l2}), \xi(s_{r1}))$. By induction hypothesis, $\xi(s_{l1}) \xrightarrow{\sigma',\Gamma'} \xi(s_{l2})$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$ by the rule $||_1$.
   - Case $||_2$: same.
   - Case $||_3$: similar proof. This time $\alpha(\sigma) \in \Delta$ and the induction hypothesis is used twice (for l and r).

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|:_1$: $s_1 = (|:_\circ, \bot, \bot)$ and $s_2 = (|:_\circ, v, ss_2)$ with $\mathit{init}(b[\overrightarrow{V}]) \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} ss_2$ and $v \in X$. We have $\xi(s_1) = (|:_\circ, \bot, \bot)$ with $\xi(s_1).\mathit{val} = \overrightarrow{V'}$. By Lemma 15, $\xi(\mathit{init}(b[\overrightarrow{V}])) = \mathit{init}(b[\overrightarrow{V'}])$. By cases on X:
     - If $X = V_i \in \overrightarrow{V}$. Let $v' = \rho_i(v)$. Then $v' \in V'_i = \rho_i(V_i)$ with $V'_i \in \overrightarrow{V'}$. By induction hypothesis, $\mathit{init}(b[\overrightarrow{V'}]) \xrightarrow{\sigma',(\![x:=v']\!)\lhd\Gamma'} \xi(ss_2)$. We have $\xi(s_2) = (|:_\circ, v', \xi(ss_2))$. Thus by the rule $|:_1$, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
     - If $X = W \notin \overrightarrow{V}$, then we still have $v \in W$. By induction hypothesis, $\mathit{init}(b[\overrightarrow{V'}]) \xrightarrow{\sigma',(\![x:=v]\!)\lhd\Gamma'} \xi(ss_2)$. We have $\xi(s_2) = (|:_\circ, v, \xi(ss_2))$. Thus by the rule $|:_1$, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
   - Case $|:_2$: $s_1 = (|:_\circ, v, ss_1)$ and $s_2 = (|:_\circ, v, ss_2)$ with $ss_1 \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} ss_2$ and $v \neq \bot$. By cases on$X$:
     - If $X = V_i \in \overrightarrow{V}$. Let $v' = \rho_i(v)$. We have $\xi(s_1) = (|:_\circ, v', \xi(ss_1))$ and $\xi(s_2) = (|:_\circ, v', \xi(ss_2))$. By induction hypothesis, $\xi(ss_1) \xrightarrow{\sigma',(\![x:=v']\!)\lhd\Gamma'} \xi(ss_2)$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
     - If $X = W \notin \overrightarrow{V}$.
       We have $\xi(s_1) = (|:_\circ, v, \xi(ss_1))$ and $\xi(s_2) = (|:_\circ, v, \xi(ss_2))$. By induction hypothesis, $\xi(ss_1) \xrightarrow{\sigma',(\![x:=v]\!)\lhd\Gamma'} \xi(ss_2)$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By $|||:_1$, the only rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$, $s_1 = (|||:_\circ, f)$ and $s_2 = (|||:_\circ, f \Lleftarrow \{v \mapsto ss_2\})$ with $f(v) \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} ss_2$. By cases on X:

   - If $X = V_k \in \overrightarrow{V}$. Let $v' = \rho_k(v)$. We have $\xi(s_1) = (|||:_\circ, f')$ with $\mathit{dom}(f') = \rho_k(\mathit{dom}(f))$ and for all $w \in \mathit{dom}(f)$, $f'(\rho_k(w)) = \xi(f(w))$. By induction hypothesis, $\xi(f(v)) = f'(v') \xrightarrow{\sigma',(\![x:=v']\!)\lhd\Gamma'} \xi(ss_2)$. By the rule $|||:_1$, we have $(|||:_\circ, f') \xrightarrow{\sigma',\Gamma'} (|||:_\circ, f' \Lleftarrow \{v' \mapsto \xi(ss_2)\})$. We have $\xi(s_2) = (|||:_\circ, f' \Lleftarrow \{v' \mapsto \xi(ss_2)\})$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.
   - If $X = W \notin \overrightarrow{V}$. We have $\xi(s_1) = (|||:_\circ, f')$ with $\mathit{dom}(f') = \mathit{dom}(f)$ and for all $w \in \mathit{dom}(f)$, $f'(w) = \xi(f(w))$. By induction hypothesis, $\xi(f(v)) = f'(v) \xrightarrow{\sigma',(\![x:=v]\!)\lhd\Gamma'} \xi(ss_2)$.

By the rule $|||:_1$, we have $(|||:_\circ, f') \xrightarrow{\sigma',\Gamma'} (|||:_\circ, f' \lhd\!\!- \{v \mapsto \xi(ss_2)\})$. We have $\xi(s_2) = (|||:_\circ, f' \lhd\!\!- \{v \mapsto \xi(ss_2)\})$. Thus, $\xi(s_1) \xrightarrow{\sigma',\Gamma'} \xi(s_2)$.

□

B.4 Proofs of Section A.5

Lemma 25. *Let $a[\overrightarrow{T}]$ be a PASTD. For any parameter values $\overrightarrow{V}$ and $\overrightarrow{V'}$ such that $\gamma(\overrightarrow{V}) \leq_k \gamma(\overrightarrow{V'})$, $init(a[\overrightarrow{V}]) \preceq init(a[\overrightarrow{V'}])$.*

Proof. By Lemmas 12 and 15. □

Lemma 26. *For any IPASTD states $s, s' \in \mathcal{Q}^\sharp$ such that $s \sqsubseteq s'$, if $final(s)$ then $final(s')$.*

Proof. Same proof as for Lemma 13. □

Lemma 27. *For any IPASTD states $s, s' \in \mathcal{Q}^\sharp$ such that $s \preceq s'$, if $final(s)$ then $final(s')$.*

Proof. By Lemmas 26 and 16 (which are easily generalized to $\mathcal{Q}^\sharp$). □

Lemma 28. *Let $a[\overrightarrow{T}]$ be a PASTD and $\overrightarrow{V}$ a parameter value. Let $q \in Abinit(a[\overrightarrow{V}])$. For all value $\overrightarrow{V'}$ such that $|\overrightarrow{V}| \leq_k |\overrightarrow{V'}|$, there exists $q' \in \mathcal{Q}^\sharp$ such that $q \preceq q'$ and $q' \in Abinit(a[\overrightarrow{V'}])$.*

Proof. Take $q' \in \mathcal{Q}^\sharp$ such that q' has the same state expression (modulo renaming) and same PASTD as q but with $q'.val = \overrightarrow{V'}$. This is possible by definition of $\mathcal{Q}^\sharp$, which allows the domain of the function of the quantified interleaving operator to be partial. □

Lemma 29. *For any $s_1, s_1', s_2 \in \mathcal{Q}^\sharp$, any event σ and any environment $\Gamma = (\![x_1, ..., x_n := v_1, ..., v_n]\!)$, if $s_1 \preceq s_1'$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, then there exists s_2' such that $s_2 \preceq s_2'$ and $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$ such that ξ is the rename function defined by $\rho_1, ..., \rho_k$ and $s_1 \sqsubseteq \xi(s_1')$ and $s_2 \sqsubseteq \xi(s_2')$, where $\sigma' = \sigma[v_1, ..., v_n := v_1', ..., v_n']$ and $\Gamma' = (\![x_1, ..., x_n := v_1', ..., v_n']\!)$ with $v_i' = \rho_j(v_i)$ if $\exists j \cdot v_i \in P_j$ or $v_i' = v_i$ else.*

Proof. The proof is done by structural induction on $s_1.pa = s_2.pa = s_1'.pa = a[\overrightarrow{T}]$ and is similar to the one for Lemma 17 (modulo renaming). We detail only the new cases defined in Definition 39.

1. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_{1a}: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_2, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\delta(n_1, n_2, \sigma'', final?)$, $final? \Rightarrow final(ss_1)$, $\sigma''[\Gamma] = \sigma$ and $ss_2 \in Abinit(\nu(n_2)[\overrightarrow{V}])$. Let $s_1'.val = \overrightarrow{V'}$. We have $s_1' = (\mathsf{aut}_\circ, n_1, ss_1')$ with $ss_1 \sqsubseteq \xi(ss_1')$. Take $s_2' = (\mathsf{aut}_\circ, n_2, init(\nu(n_2)[\overrightarrow{V'}]))$. By Definition 38, we have $ss_2 \sqsubseteq init(\nu(n_2)[\overrightarrow{V}])$ and, by Lemma 25, $init(\nu(n_2)[\overrightarrow{V}]) \preceq init(\nu(n_2)[\overrightarrow{V'}])$. Thus, $ss_2 \preceq ss_2'$ and $s_2 \preceq s_2'$. By Lemma 27, $final? \Rightarrow final(ss_1')$. Moreover, $\sigma''[\Gamma'] = \sigma[v_1, ..., v_n := v_1', ..., v_n'] = \sigma'$. Consequently, by the inference rule aut_{1a}, $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$.

2. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_{1a}$: $s_1 = (\star_\circ, started?, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $final(ss_1) \vee \neg started?$, $q \xrightarrow{\sigma,\Gamma} ss_2$ and $q \in Abinit(b[\overrightarrow{V}])$. Let $s_1' \in \mathcal{Q}^\sharp$ such that $s_1 \preceq s_1'$ with $s_1'.val = \overrightarrow{V'}$. We have $s_1' = (\star_\circ, started?, ss_1')$ with $ss_1 \preceq ss_1'$. Take $q' \in \mathcal{Q}^\sharp$ such that $q \preceq q'$ and $q' \in Abinit(b[\overrightarrow{V'}])$ thanks to Lemma 28. By induction hypothesis, there exists $ss_2' \in \mathcal{Q}^\sharp$ such that $ss_2 \preceq ss_2'$ and $q' \xrightarrow{\sigma',\Gamma'} ss_2'$. Let $s_2' = (\star_\circ, true, ss_2')$. We have $s_2 \preceq s_2'$. By Lemma 27, $final(ss_1') \vee \neg started?$. Thus, by the inference rule $\star_{1a}$, $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$.

3. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|_{1a}$: $s_1 = (|_\circ, \bot, \bot)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $q \xrightarrow{\sigma,\Gamma} ss_2$ and $q \in Abinit(l[\overrightarrow{V}])$. Let $s_1' \in \mathcal{Q}^\sharp$ such that $s_1 \preceq s_1'$ with $s_1'.val = \overrightarrow{V'}$. We have $s_1' = (|_\circ, \bot, \bot)$. Take $q' \in \mathcal{Q}^\sharp$ such that $q \preceq q'$ and $q' \in Abinit(l[\overrightarrow{V'}])$ thanks to Lemma 28. By induction hypothesis, there exists $ss_2' \in \mathcal{Q}^\sharp$ such that $ss_2 \preceq ss_2'$ and $q' \xrightarrow{\sigma',\Gamma'} ss_2'$. Let $s_2' = (|_\circ, \mathsf{left}, ss_2')$. We have $s_2 \preceq s_2'$. Finally, by the inference rule $\star_{1a}$, $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$.
   - Case $|_{2a}$: same as previous.

4. Inductive case: $a[\overrightarrow{T}] = (|\!:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|\!:_{1a}$: $s_1 = (|\!:_\circ, \bot, \bot)$ and $s_2 = (|\!:_\circ, v, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $q \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} ss_2$, $q \in Abinit(b[\overrightarrow{V}])$ and $v \in X$. Let $s_1' \in \mathcal{Q}^\sharp$ such that $s_1 \preceq s_1'$ with $s_1'.val = \overrightarrow{V'}$. We have $s_1' = (|\!:_\circ, \bot, \bot)$. Take $q' \in \mathcal{Q}^\sharp$ such that $q \preceq q'$ and $q' \in Abinit(b[\overrightarrow{V'}])$ thanks to Lemma 28. By cases on X:
     - If $X = T_i$, then $v \in V_i$. Let $v' = \rho_i^{-1}(v)$. By induction hypothesis, there exists $ss_2' \in \mathcal{Q}^\sharp$ such that $ss_2 \preceq ss_2'$ and $q' \xrightarrow{\sigma',(\![x:=v']\!)\lhd\Gamma'} ss_2'$. Let $s_2' = (|\!:_\circ, v', ss_2')$. Then, $s_2 \preceq s_2'$. Moreover, $v' \in \rho_i^{-1}(V_i) \subseteq V_i'$. Thus, by the rule $|\!:_{1a}$, $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$.
     - If $X = W \notin \overrightarrow{T}$, then $v \in W$. By induction hypothesis, there exists $ss_2' \in \mathcal{Q}^\sharp$ such that $ss_2 \preceq ss_2'$ and $q' \xrightarrow{\sigma',(\![x:=v]\!)\lhd\Gamma'} ss_2'$. Let $s_2' = (|\!:_\circ, v, ss_2')$. We have $s_2 \preceq s_2'$ and $s_1' \xrightarrow{\sigma',\Gamma'} s_2'$.

□

Theorem 9. Let $a[\overrightarrow{T}]$ be a PASTD without free variables in events, with $\mathcal{S}_a^\sharp$ the corresponding TS from Definition 41. $\mathcal{S}_a^\sharp$ is monotone wrt $\sqsubseteq$ and $\preceq$.

Proof. Let s_1, s_1', s_2 states of $\mathcal{S}_a^\sharp$ such that $s_1 \preceq s_1'$ and $s_1 \to s_2$. There is an event σ such that $s_1 \xrightarrow{\sigma} s_2$. By the only inference rule env, we have $s_1 \xrightarrow{\sigma,(\![]\!)} s_2$. Then, by Lemma 29 there exists $s_2' \in \mathcal{Q}^\sharp$ such that $s_2 \preceq s_2'$ and $s_1' \xrightarrow{\sigma',(\![]\!)} s_2'$, with σ' a renaming of σ. Thus, by the rule env, we have $s_1' \xrightarrow{\sigma'} s_2'$. As s_2' is a state of $\mathcal{S}_a^\sharp$ too, we can conclude. Similar proof for $\preceq$. □

Lemma 30. *Let $s_1, s_2 \in \mathcal{Q}^\sharp$. Consider the transition rules from Definition 39. If $s_1 \xrightarrow{\sigma,\Gamma} s_2$ and $s_1 \in \mathcal{Q}$, then there exists $s_3 \in \mathcal{Q}$ such that $s_1 \xrightarrow{\sigma,\Gamma} s_3$ and $s_2 \sqsubseteq s_3$.*

Proof. Let $s_1, s_2 \in \mathcal{Q}^\sharp$, σ an event and Γ an environment such that $s_1 \xrightarrow{\sigma,\Gamma} s_2$ and $s_1 \in \mathcal{Q}$. By structural induction on $s_1.pa = s_2.pa = a[\overrightarrow{T}]$.

1. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_{1a}: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_2, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\delta(n_1, n_2, \sigma', final?)$, $final? \Rightarrow \mathit{final}(ss_1)$, $\sigma'[\Gamma] = \sigma$ and $ss_2 \in \mathit{Abinit}(\nu(n_2)[\overrightarrow{V}])$. Let $ss_3 = \mathit{init}(\nu(n_2)[\overrightarrow{V}])$ and $s_3 = (\mathsf{aut}_\circ, n_2, ss_3)$. We have $ss_2 \sqsubseteq ss_3$ and $s_2.val = s_3.val$. Thus, $s_2 \sqsubseteq s_3$. By Lemma 10, $ss_3 \in \mathcal{Q}$. Thus, $s_3 \in \mathcal{Q}$. By inference rule aut_{1a}, $s_1 \xrightarrow{\sigma,\Gamma} s_3$.
   - Case aut_2: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_1, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$ and $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. If $s_1 \in \mathcal{Q}$, then $ss_1 \in \mathcal{Q}$. By induction hypothesis, there exists $ss_3 \in \mathcal{Q}$ such that $ss_1 \xrightarrow{\sigma,\Gamma} ss_3$ and $ss_2 \sqsubseteq ss_3$. Let $s_3 = (\mathsf{aut}_\circ, n_1, ss_3)$. We have $s_3 \in \mathcal{Q}$ and $s_2 \sqsubseteq s_3$. By inference rule aut_2, $s_1 \xrightarrow{\sigma,\Gamma} s_3$.

2. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_{1a}$: $s_1 = (\star_\circ, started?, ss_1)$ and $s_2 = (\star_\circ, \mathrm{true}, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\mathit{final}(ss_1) \vee \neg started?$, $q \xrightarrow{\sigma,\Gamma} ss_2$ and $q \in \mathit{Abinit}(b[\overrightarrow{V}])$. Let $q' = \mathit{init}(b[\overrightarrow{V}])$. We have $q' \in \mathcal{Q}$ and $q \sqsubseteq q'$. By compatibility, there exists $ss_2' \in \mathcal{Q}^\sharp$ such that $q' \xrightarrow{\sigma,\Gamma} ss_2'$ and $ss_2 \sqsubseteq ss_2'$. By induction hypothesis, there exists $ss_3 \in \mathcal{Q}$ such that $q' \xrightarrow{\sigma,\Gamma} ss_3$ and $ss_2' \sqsubseteq ss_3$. Let $s_3 = (\star_\circ, \mathrm{true}, ss_3)$. We have $s_3 \in \mathcal{Q}$ and $s_2 \sqsubseteq s_2' \sqsubseteq s_3$. By inference rule $\star_{1a}$, $s_1 \xrightarrow{\sigma,\Gamma} s_3$.
   - Case $\star_2$: similar to case aut_2.

3. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Cases $|_{1a}$ and $|_{2a}$: similar to case $\star_{1a}$.
   - Cases $|_3$ and $|_4$: similar to case aut_2.

4. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Cases $||_1$, $||_2$ and $||_3$: similar to case aut_2.

5. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|:_{1a}$: similar to case $\star_{1a}$.
   - Case $|:_2$: similar to case aut_2.

6. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

- Case $|||:_1$: $s_1 = (|||:_\circ, f))$ and $s_2 = (|||:_\circ, f \Leftarrow \{v \mapsto ss_2)$. Similar to case aut_2, with induction hypothesis on ss_2. Note that if $s_1 \in \mathcal{Q}$, then for all $v' \in dom(f)$, $f(v') \in \mathcal{Q}$.

□

Lemma 31. *Let us denote by $\rightarrow_1$ the transition relation from Definition 25 and by $\rightarrow_2$ the transition relation from Definition 39. Let $s_1, s_2 \in \mathcal{Q}$. If $s_1 \xrightarrow{\sigma,\Gamma}_2 s_2$, then $s_1 \xrightarrow{\sigma,\Gamma}_1 s_2$.*

Proof. Let $s_1, s_2 \in \mathcal{Q}$, σ an event and Γ an environment such that $s_1 \xrightarrow{\sigma,\Gamma}_2 s_2$. By structural induction on $s_1.pa = s_2.pa = a[\overrightarrow{T}]$, let us show that $s_1 \xrightarrow{\sigma,\Gamma}_1 s_2$. We detail only the new rules, as the other cases are obvious.

1. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma}_2 s_2$:

   - Case aut_{1a}: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_2, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\delta(n_1, n_2, \sigma', final?)$, $final? \Rightarrow final(ss_1)$, $\sigma'[\Gamma] = \sigma$ and $ss_2 \in Abinit(\nu(n_2)[\overrightarrow{V}])$. As $s_2 \in \mathcal{Q}$, then $ss_2 \in \mathcal{Q}$. Thus, $ss_2 = init(\nu(n_2)[\overrightarrow{V}])$. By the inference rule aut_1, $s_1 \xrightarrow{\sigma,\Gamma}_1 s_2$.

2. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $\star_{1a}$: $s_1 = (\star_\circ, started?, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $final(ss_1) \vee \neg started?$, $q \xrightarrow{\sigma,\Gamma}_2 ss_2$ and $q \in Abinit(b[\overrightarrow{V}])$. Let $q' = init(b[\overrightarrow{V}])$. We have $q' \in \mathcal{Q}$ and $q \sqsubseteq q'$. By compatibility, there exists $ss'_2 \in \mathcal{Q}^\sharp$ such that $q' \xrightarrow{\sigma,\Gamma}_2 ss'_2$ and $ss_2 \sqsubseteq ss'_2$. As $q.val = q'.val = \overrightarrow{V}$, then $ss'_2.val = \overrightarrow{V}$. We have $ss_2.val = \overrightarrow{V} = ss'_2.val$, $ss_2 \in \mathcal{Q}$ and $ss_2 \sqsubseteq ss'_2$. Thus, $ss_2 = ss'_2$. As a consequence, $init(b[\overrightarrow{V}]) \xrightarrow{\sigma,\Gamma}_2 ss_2$ and by induction hypothesis, $init(b[\overrightarrow{V}]) \xrightarrow{\sigma,\Gamma}_1 ss_2$. Thus, by the inference rule $\star_1$, $s_1 \xrightarrow{\sigma,\Gamma}_1 s_2$.

3. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Cases $|_{1a}$ and $|_{2a}$: similar to case $\star_{1a}$.

4. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b)[\overrightarrow{T}]$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $|:_{1a}$: similar to case $\star_{1a}$.

□

Lemma 18. Let $a[\overrightarrow{T}]$ be a PASTD, $\mathcal{S}_a = (Q, \rightarrow)$ the TS derived from Definition 32 with initial states I, and $\mathcal{S}^\sharp_a = (Q', \rightarrow')$ the TS from Definition 41 with initial states I'. Then, for all path $s'_0 \rightarrow' ... \rightarrow' s'_n$ in $\mathcal{S}^\sharp_a$ with $s'_0 \in I'$, there exists a path $s_0 \rightarrow ... \rightarrow s_n$ in $\mathcal{S}_a$ with $s_0 \in I$ such that $s'_i \leq s_i$ for all $i \leq n$.

Proof. We have $Q \subseteq Q' \subseteq \mathcal{Q}^\sharp$ and $\rightarrow \subseteq \rightarrow'$. Let $s'_0 \rightarrow' ... \rightarrow' s'_n$ be a path in $\mathcal{S}^\sharp_a$ with $s'_0 \in I'$. Let $\overrightarrow{V} = s'_0.val$. By induction on the length n of the path $s'_0 \rightarrow' ... \rightarrow' s'_n$, let us show that there exists a path $s_0 \rightarrow ... \rightarrow s_n$ in $\mathcal{S}_a$ with $s_0 \in I$ such that $s'_i \sqsubseteq s_i$ for all $i \leq n$.

- Base case: $n = 1$, $s'_0 \to' s'_1$. Let $s_0 = init(a[\overrightarrow{V}])$. We have $s_0 \in I$ and $s'_0 \sqsubseteq s_0$. By monotony (Theorem 9), there exists $q \in Q'$ such that $s_0 \to' q$ and $s'_1 \sqsubseteq q$. As $s_0 \in \mathcal{Q}$, take $s_1 \in \mathcal{Q}$ such that $s_0 \to' s_1$ and $q \sqsubseteq s_1$ thanks to Lemma 30. We have $s_1 \in Q$ and $s'_1 \sqsubseteq s_1$. As $s_0, s_1 \in Q$ and $s_0 \to' s_1$, then $s_0 \to s_1$ by Lemma 31.

- Inductive case: let $s'_0 \to' ... \to' s'_{n+1}$ be a path in $\mathcal{S}^\sharp_a$ with $s'_0 \in I'$ and suppose that there exists a path $s_0 \to ... \to s_n$ in $\mathcal{S}_a$ with $s_0 \in I$ such that $s'_i \sqsubseteq s_i$ for all $i \leq n$. Let us show that there is a path of length $n + 1$ in $\mathcal{S}_a$. By monotony (Theorem 9), there exists $q \in Q'$ such that $s_n \to' q$ and $s'_{n+1} \sqsubseteq q$. As $s_n \in \mathcal{Q}$, take $s_{n+1} \in \mathcal{Q}$ such that $s_n \to' s_{n+1}$ and $q \sqsubseteq s_{n+1}$ thanks to Lemma 30. We have $s_{n+1} \in Q$ and $s'_{n+1} \sqsubseteq s_{n+1}$. As $s_n, s_{n+1} \in Q$ and $s_n \to' s_{n+1}$, then $s_n \to s_{n+1}$ by Lemma 31.

□

B.5 Proofs of Section A.6

Lemma 32. *Let $a[\overrightarrow{T}]$ be a PASTD and $\overrightarrow{V}$ a parameter value. Let $s \in \mathcal{Q}^\sharp$ such that $s \in Abinit(a[\overrightarrow{V}])$. Then, for all $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$ and with $q.val = \overrightarrow{W}$, we have $q \in Abinit(a[\overrightarrow{W}])$.*

Proof. Let $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$ with $q.val = \overrightarrow{W}$. By Definition 38, $s \sqsubseteq init(a[\overrightarrow{V}])$. By transitivity, $q \sqsubseteq init(a[\overrightarrow{V}])$. By definition of *init*, $q \sqsubseteq init(a[\overrightarrow{W}])$ (easy proof by induction on $a[\overrightarrow{T}]$). □

Lemma 20. Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{T}]$, there exists $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$ and $\gamma(q) \leq_k c$.

Proof. Let $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{V}]$ with $\overrightarrow{V} = s.val$. Let $c = \mu(a[\overrightarrow{T}])$. By structural induction on $s.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. We have $s = (\mathsf{elem}_\circ)$. Take $q = (\mathsf{elem}_\circ)$ with $\gamma(q) = (0, ..., 0)$. Thus $\gamma(q) = c$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. We have $s = (\mathsf{aut}_\circ, n, ss)$ with $ss.pa = \nu(n)[\overrightarrow{T}]$. We have $c = sup(\{\mu(\nu(n')) \mid n' \in N\})$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(\nu(n)[\overrightarrow{T}])$. Take $q = (\mathsf{aut}_\circ, n, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k \mu(\nu(n)[\overrightarrow{T}]) \leq_k c$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. We have $c = \mu(a[\overrightarrow{T}]) = \mu(b[\overrightarrow{T}])$. If $s = (\star_\circ, \text{false}, \bot)$, then take $q = (\star_\circ, \text{false}, \bot)$ with $\gamma(q) = (0, ..., 0) \leq_k c$. Else, we have $s = (\star_\circ, \text{true}, ss)$ with $ss.pa = b[\overrightarrow{T}]$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$. Take $q = (\star_\circ, \text{true}, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k \mu(b[\overrightarrow{T}]) = c$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. We have $c = sup(\mu(l[\overrightarrow{T}]), \mu(r[\overrightarrow{T}]))$. If $s = (|_\circ, \bot, \bot)$, then take $q = (|_\circ, \bot, \bot)$ with $\gamma(q) = (0, ..., 0) \leq_k c$. Else, we have $s = (|_\circ, side, ss)$ with $ss.pa = l[\overrightarrow{T}]$ (or $ss.pa = r[\overrightarrow{T}]$). By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that

$qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(l[\overrightarrow{T}])$ (or $\gamma(qq) \leq_k \mu(r[\overrightarrow{T}])$). Take $q = (|_\circ, side, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c$.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $c = \mu(l[\overrightarrow{T}]) + \mu(r[\overrightarrow{T}])$. We have $s = (||_\circ, s_l, s_r)$ with $s_l.pa = l[\overrightarrow{T}]$ and $s_r.pa = r[\overrightarrow{T}]$. By induction hypothesis, there exist $q_l, q_r \in \mathcal{Q}^\sharp$ such that $q_l \sqsubseteq s_l$, $q_r \sqsubseteq s_r$, $\gamma(q_l) \leq_k \mu(l[\overrightarrow{T}])$ and $\gamma(q_r) \leq_k \mu(r[\overrightarrow{T}])$. Let $\overrightarrow{W} = q_l.val \cup q_r.val$. By Lemma 19, take $q'_l, q'_r \in \mathcal{Q}^\sharp$ such that $q_l \sqsubseteq q'_l \sqsubseteq s_l$, $q_r \sqsubseteq q'_r \sqsubseteq s_r$ and $q'_l.val = q'_r.val = \overrightarrow{W}$. Take $q = (||_\circ, q'_l, q'_r) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) \leq_k \gamma(q_l) + \gamma(q_r) \leq_k c$.

6. Inductive case: $a[\overrightarrow{T}] = (|:, n, x, X, b[\overrightarrow{T}])$. If $s = (|:_\circ, \bot, \bot)$, then take $q = (|:_\circ, \bot, \bot)$ with $\gamma(q) = (0, ..., 0) \leq_k c$. Else, we have $s = (|:_\circ, v, ss)$ with $ss.pa = b[\overrightarrow{T}]$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$. By cases on X:

    - If X is not a parameter, then take $q = (|:_\circ, v, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c = \mu(b[\overrightarrow{T}])$.
    - If $X = T_i$ is the i-th parameter, then let $\overrightarrow{W} = qq.val \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. By Lemma 19, take $qq' \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq qq' \sqsubseteq ss$ and $qq'.val = \overrightarrow{W}$. Let $q = (|:, v, qq')$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq') \leq_k \gamma(qq) + (0, ..., 1, ..., 0) \leq_k \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0) = c$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b[\overrightarrow{T}])$. We have $s = (|||:_\circ, f)$ with for all $ss \in ran(f)$, $ss.pa = b[\overrightarrow{T}]$. By induction hypothesis, for all $ss_i \in ran(f)$, let $qq_i \in \mathcal{Q}^\sharp$ such that $qq_i \sqsubseteq ss_i$ and $\gamma(qq_i) \leq_k \mu(b[\overrightarrow{T}])$. By cases on X:

    - If X is not a parameter, then let $\overrightarrow{W} = \bigcup_i qq_i.val \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k \sum_i \gamma(qq_i) = |X| \cdot \gamma(qq_1)$. By Lemma 19, for each qq_i take $qq'_i \in \mathcal{Q}^\sharp$ such that $qq_i \sqsubseteq qq'_i \sqsubseteq ss_i$ and $qq'_i.val = \overrightarrow{W}$. Take $q = (|||:_\circ, f') \in \mathcal{Q}^\sharp$ such that $dom(f') = dom(f)$, $f'(f^{-1}(ss_i)) = qq'_i$, $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = |X| \cdot \gamma(qq_1) \leq_k |X| \cdot \mu(b[\overrightarrow{T}]) = c$.
    - If $X = T_i$ is the i-th parameter, then choose a value $v \in dom(f)$ and let $ss = f(v)$. By induction hypothesis, let $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$. Let $\overrightarrow{W} = qq.val \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. By Lemma 19, take $qq' \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq qq' \sqsubseteq ss$ and $qq'.val = \overrightarrow{W}$. Let $q = (|||:, \{v \mapsto qq'\})$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = |\overrightarrow{W}| \leq_k \gamma(qq) + (0, ..., 1, ..., 0) \leq_k \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0) = c$.

□

Lemma 21. Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{T}]$, if *final*$(s)$, then there exists $q \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq s$, $\gamma(q) \leq_k c$ and *final*(q).

Proof. Let $s \in \mathcal{Q}^\sharp$ such that $s.pa = a[\overrightarrow{V}]$ with $\overrightarrow{V} = s.val$ and *final*(s). Let $c = \mu(a[\overrightarrow{T}])$. By structural induction on $s.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. We have $s = (\mathsf{elem}_\circ)$. Take $q = (\mathsf{elem}_\circ)$ with $\gamma(q) = (0, ..., 0)$. Thus $\gamma(q) = c$. Moreover, we have *final*(q).

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, \mathit{name}, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. We have $s = (\mathsf{aut}_\circ, n, ss)$ with $ss.pa = \nu(n)[\overrightarrow{T}]$. We have $c = \mathit{sup}(\{\mu(\nu(n')) \mid n' \in N\})$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(\nu(n)[\overrightarrow{T}])$ and $\mathit{final}(ss) \implies \mathit{final}(qq)$. Take $q = (\mathsf{aut}_\circ, n, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k \mu(\nu(n)[\overrightarrow{T}]) \leq_k c$. As $\mathit{final}(s)$, either $n \in DF$ and $\mathit{final}(q)$, or $\mathit{final}(ss)$ and $\mathit{final}(qq)$, thus $\mathit{final}(q)$.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. We have $c = \mu(a[\overrightarrow{T}]) = \mu(b[\overrightarrow{T}])$. If $s = (\star_\circ, \text{false}, \bot)$, then take $q = (\star_\circ, \text{false}, \bot)$ with $\gamma(q) = (0, ..., 0) \leq_k c$. Moreover, we have $\mathit{final}(q)$. Else, we have $s = (\star_\circ, \text{true}, ss)$ with $ss.pa = b[\overrightarrow{T}]$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$ and $\mathit{final}(ss) \implies \mathit{final}(qq)$. Take $q = (\star_\circ, \text{true}, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k \mu(b[\overrightarrow{T}]) = c$. As $\mathit{final}(ss)$, we have $\mathit{final}(qq)$, thus $\mathit{final}(q)$.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. We have $c = \mathit{sup}(\mu(l[\overrightarrow{T}]), \mu(r[\overrightarrow{T}]))$.

   - If $s = (|_\circ, \bot, \bot)$, then $\mathit{final}(\mathit{init}(l[\overrightarrow{V}])) \vee \mathit{final}(\mathit{init}(r[\overrightarrow{V}]))$.
     - Case $\mathit{final}(\mathit{init}(l[\overrightarrow{V}]))$: let $ss = \mathit{init}(l[\overrightarrow{V}])$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(l[\overrightarrow{T}])$ and $\mathit{final}(qq)$. Let $\overrightarrow{W} = qq.val$. By Lemma 32, $qq \in \mathit{Abinit}(l[\overrightarrow{W}])$. Thus, by Lemma 26, $\mathit{final}(\mathit{init}(l[\overrightarrow{W}]))$. Take $q = (|_\circ, \bot, \bot)$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $\mathit{final}(q)$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c$.
     - Case $\mathit{final}(\mathit{init}(r[\overrightarrow{V}]))$: similar.
   - Else, we have $s = (|_\circ, \mathit{side}, ss)$ with $ss.pa = l[\overrightarrow{T}]$ (or $ss.pa = r[\overrightarrow{T}]$). As $\mathit{final}(s)$, then $\mathit{final}(ss)$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$ and $\gamma(qq) \leq_k \mu(l[\overrightarrow{T}])$ (or $\gamma(qq) \leq_k \mu(r[\overrightarrow{T}])$) and $\mathit{final}(qq)$. Take $q = (|_\circ, \mathit{side}, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c$. And as $\mathit{final}(qq)$, we have $\mathit{final}(q)$.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. We have $c = \mu(l[\overrightarrow{T}]) + \mu(r[\overrightarrow{T}])$. We have $s = (||_\circ, s_l, s_r)$ with $s_l.pa = l[\overrightarrow{T}]$ and $s_r.pa = r[\overrightarrow{T}]$. Moreover, as $\mathit{final}(s)$, then $\mathit{final}(s_l) \wedge \mathit{final}(s_r)$. By induction hypothesis, there exist $q_l, q_r \in \mathcal{Q}^\sharp$ such that $q_l \sqsubseteq s_l$, $q_r \sqsubseteq s_r$, $\gamma(q_l) \leq_k \mu(l[\overrightarrow{T}])$, $\gamma(q_r) \leq_k \mu(r[\overrightarrow{T}])$, $\mathit{final}(q_l)$ and $\mathit{final}(q_r)$. Let $\overrightarrow{W} = q_l.val \cup q_r.val$. By Lemma 19, take $q'_l, q'_r \in \mathcal{Q}^\sharp$ such that $q_l \sqsubseteq q'_l \sqsubseteq s_l$, $q_r \sqsubseteq q'_r \sqsubseteq s_r$ and $q'_l.val = q'_r.val = \overrightarrow{W}$. Take $q = (||_\circ, q'_l, q'_r) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) \leq_k \gamma(q_l) + \gamma(q_r) \leq_k c$. By Lemma 26, $\mathit{final}(q'_l)$ and $\mathit{final}(q'_r)$. Thus, $\mathit{final}(q)$.

6. Inductive case: $a[\overrightarrow{T}] = (|\colon, n, x, X, b[\overrightarrow{T}])$.

   - If $s = (|\colon_\circ, \bot, \bot)$, then $\mathit{final}(\mathit{init}(b[\overrightarrow{V}]))$. Let $ss = \mathit{init}(b[\overrightarrow{V}])$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$ and $\mathit{final}(qq)$. Let $\overrightarrow{W} = qq.val$. By Lemma 32, $qq \in \mathit{Abinit}(b[\overrightarrow{W}])$. Thus, we have $\mathit{final}(\mathit{init}(b[\overrightarrow{W}]))$ by Lemma 26. Take $q = (|\colon_\circ, \bot, \bot)$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $\mathit{final}(q)$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c$ (if X is a parameter or not).

- Else, we have $s = (|:_\circ, v, ss)$ with $ss.pa = b[\overrightarrow{T}]$ and $\mathit{final}(ss)$. By induction hypothesis, there exists $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$ and $\mathit{final}(qq)$. By cases on X:
  - If X is not a parameter, then take $q = (|:_\circ, v, qq) \in \mathcal{Q}^\sharp$ such that $q.pa = s.pa$ and $q.val = qq.val$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq) \leq_k c = \mu(b[\overrightarrow{T}])$. And as $\mathit{final}(qq)$, we have $\mathit{final}(q)$.
  - If $X = T_i$ is the i-th parameter, then let $\overrightarrow{W} = qq.val \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. By Lemma 19, take $qq' \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq qq' \sqsubseteq ss$ and $qq'.val = \overrightarrow{W}$. Let $q = (|:, v, qq')$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = \gamma(qq') \leq_k \gamma(qq) + (0, ..., 1, ..., 0) \leq_k \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0) = c$. And as $\mathit{final}(qq)$, then $\mathit{final}(qq')$ by Lemma 26. Thus, $\mathit{final}(q)$.

7. Inductive case: $a[\overrightarrow{T}] = (|||:, n, x, X, b[\overrightarrow{T}])$. We have $s = (|||:_\circ, f)$ where for all $ss \in ran(f)$, $ss.pa = b[\overrightarrow{T}]$. By induction hypothesis, for all $ss_i \in ran(f)$, let $qq_i \in \mathcal{Q}^\sharp$ such that $qq_i \sqsubseteq ss_i$, $\gamma(qq_i) \leq_k \mu(b[\overrightarrow{T}])$ and $\mathit{final}(ss_i) \implies \mathit{final}(qq_i)$. By cases on X:

   - If X is not a parameter, then for all $v \in X$, $\mathit{final}(f(v))$. Let $\overrightarrow{W} = \bigcup_i qq_i.val \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k \sum_i \gamma(qq_i) = |X| \cdot \gamma(qq_1)$. By Lemma 19, for each qq_i take $qq_i' \in \mathcal{Q}^\sharp$ such that $qq_i \sqsubseteq qq_i' \sqsubseteq ss_i$ and $qq_i'.val = \overrightarrow{W}$. Take $q = (|||:_\circ, f') \in \mathcal{Q}^\sharp$ such that $dom(f') = dom(f)$, $f'(f^{-1}(ss_i)) = qq_i'$, $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = |X| \cdot \gamma(qq_1) \leq_k |X| \cdot \mu(b[\overrightarrow{T}]) = c$. By Lemma 26, for all $qq_i' \in dom(f')$, $\mathit{final}(qq_i')$. Thus, $\mathit{final}(q)$.
   - If $X = T_i$ is the i-th parameter, take $v \in dom(f)$ such that $\mathit{final}(f(v))$ and let $ss = f(v)$. By induction hypothesis, let $qq \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq ss$, $\gamma(qq) \leq_k \mu(b[\overrightarrow{T}])$ and $\mathit{final}(qq)$. Let $\overrightarrow{W} = qq.val \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. By Lemma 19, take $qq' \in \mathcal{Q}^\sharp$ such that $qq \sqsubseteq qq' \sqsubseteq ss$ and $qq'.val = \overrightarrow{W}$. Let $q = (|||:, \{v \mapsto qq'\})$ with $q.pa = s.pa$ and $q.val = \overrightarrow{W}$. We have $q \sqsubseteq s$ and $\gamma(q) = |\overrightarrow{W}| \leq_k \gamma(qq) + (0, ..., 1, ..., 0) \leq_k \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0) = c$. By Lemma 26, $\mathit{final}(qq')$. Thus, $\mathit{final}(q)$.

□

Lemma 33. *Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s_1, s_2 \in \mathcal{Q}^\sharp$ such that $s_1.pa = s_2.pa = a[\overrightarrow{T}]$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, there exist $q_1, q_2 \in \mathcal{Q}^\sharp$ such that $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and $q_1 \xrightarrow{\sigma,\Gamma} q_2$ and:*

1. *$\gamma(q_1) = \gamma(q_2) \leq_k c$, and*
2. *$q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$, and*
3. *for all $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q_2' \sqsubseteq s_2$, there exists $q_1' \in \mathcal{Q}^\sharp$ such that $q_1 \sqsubseteq q_1' \sqsubseteq s_1$ and $q_1' \xrightarrow{\sigma,\Gamma} q_2'$.*

Proof. Let $s_1, s_2 \in \mathcal{Q}^\sharp$ two states, σ an event and Γ an environment such that $s_1 \xrightarrow{\sigma,\Gamma} s_2$. Let $\overrightarrow{V} = s_1.val = s_2.val$. By structural induction on $s_1.pa = s_2.pa = a[\overrightarrow{T}]$.

1. Base case: $a[\overrightarrow{T}] = (\mathsf{elem})$. There is no transition from $(\mathsf{elem}_\circ)$.

2. Inductive case: $a[\overrightarrow{T}] = (\mathsf{aut}, name, \Sigma, N, \nu, \delta, SF, DF, n_0)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}]) = sup(\{\mu(\nu(n')) \mid n' \in N\})$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case aut_{1a}: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_2, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\delta(n_1, n_2, \sigma', final?)$, $final? \Rightarrow final(ss_1)$, $\sigma'[\Gamma] = \sigma$ and $ss_2 \in Abinit(\nu(n_2)[\overrightarrow{V}])$. Let $cc_1 = \mu(ss_1.pa) = \mu(\nu(n_1))$ and $cc_2 = \mu(ss_2.pa) = \mu(\nu(n_2))$.
     - If $final?$, then $final(ss_1)$. By Lemma 21, take $qq_1 \in \mathcal{Q}^\sharp$ such that $final(qq_1)$, $\gamma(qq_1) \leq_k cc_1$ and $qq_1 \sqsubseteq ss_1$. We also have that for all $qq'_1 \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq'_1 \sqsubseteq ss_1$, $final(qq'_1)$ by Lemma 13. By Lemma 20, take $qq_2 \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq ss_2$ and $\gamma(qq_2) \leq_k cc_2$. We have $qq_2 \in Abinit(\nu(n_2)[qq_2.val])$ by Lemma 32. Let $\overrightarrow{W} = sup(qq_1.val, qq_2.val) \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k sup(cc_1, cc_2)$ by definition of sup. Take $qq'_1 \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq'_1 \sqsubseteq ss_1$ and $qq'_1.val = \overrightarrow{W}$, and $qq'_2 \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$ and $qq'_2.val = \overrightarrow{W}$, which both exist by Lemma 19. We still have $final(qq'_1)$ and $qq'_2 \in Abinit(\nu(n_2)[\overrightarrow{W}])$. Let $q_1 = (\mathsf{aut}_\circ, n_1, qq'_1)$ and $q_2 = (\mathsf{aut}_\circ, n_2, qq'_2)$. Thus, by the inference rule aut_{1a}, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
       (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k sup(cc_1, cc_2) \leq_k c$.
       (b) As $qq'_1 \sqsubseteq ss_1$ and $qq'_2 \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.
       (c) Let $q''_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q''_2 \sqsubseteq s_2$ with $q''_2.val = \overrightarrow{V'}$. We have $q''_2 = (\mathsf{aut}_\circ, n_2, qq''_2)$ with $qq'_2 \sqsubseteq qq''_2 \sqsubseteq ss_2$ and $qq''_2 \in Abinit(\nu(n_2)[\overrightarrow{V'}])$ by Lemma 32. Take $qq''_1 \in \mathcal{Q}^\sharp$ such that $qq''_1.val = \overrightarrow{V'}$ and $qq'_1 \sqsubseteq qq''_1 \sqsubseteq ss_1$, and which exists by Lemma 19. We have $final(qq''_1)$ by Lemma 13. Let $q''_1 = (\mathsf{aut}_\circ, n_1, qq''_1)$. Thus, $q_1 \sqsubseteq q''_1 \sqsubseteq s_1$. And by the rule aut_{1a}, $q''_1 \xrightarrow{\sigma,\Gamma} q''_2$.
     - If $\neg final?$, then take $qq_1 \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq ss_1$ and $\gamma(qq_1) \leq_k cc_1$, thanks to Lemma 20 instead of Lemma 21. The rest is similar to the case $final?$, without using the "final" conditions.
   - Case aut_2: $s_1 = (\mathsf{aut}_\circ, n_1, ss_1)$ and $s_2 = (\mathsf{aut}_\circ, n_1, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$ and $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. Let $cc = \mu(ss_1.pa) = \mu(\nu(n_1)[\overrightarrow{T}])$. By induction hypothesis, take $qq_1, qq_2 \in \mathcal{Q}^\sharp$ such that $qq_1 \xrightarrow{\sigma,\Gamma} qq_2$ with $\gamma(qq_1) = \gamma(qq_2) \leq_k cc$, $qq_1 \sqsubseteq ss_1$, $qq_2 \sqsubseteq ss_2$ and for all $qq'_2 \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$ there exists $qq'_1 \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq'_1 \sqsubseteq ss_1$ and $qq'_1 \xrightarrow{\sigma,\Gamma} qq'_2$. Let $\overrightarrow{W} = qq_1.val$, $q_1 = (\mathsf{aut}_\circ, n_1, qq_1)$ with $q_1.pa = a$ and $q_1.val = \overrightarrow{W}$ and $q_2 = (\mathsf{aut}_\circ, n_1, qq_2)$ with $q_2.pa = a$ and $q_2.val = \overrightarrow{W}$. By the rule aut_2, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
     (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k cc \leq_k c$.
     (b) As $qq_1 \sqsubseteq ss_1$ and $qq_2 \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.
     (c) Let $q'_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q'_2 \sqsubseteq s_2$ with $q'_2.val = \overrightarrow{V'}$. We have $q'_2 = (\mathsf{aut}_\circ, n_1, qq'_2)$ with $q'_2.pa = a$ and $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$. Take $qq'_1 \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $qq_1 \sqsubseteq qq'_1 \sqsubseteq ss_1$ and $qq'_1 \xrightarrow{\sigma,\Gamma} qq'_2$. Let $q'_1 = (\mathsf{aut}_\circ, n_1, qq'_1)$ with $q'_1.pa = a$ and $q'_1.val = \overrightarrow{V'}$. We have $q_1 \sqsubseteq q'_1 \sqsubseteq s_1$ because $qq_1 \sqsubseteq qq'_1 \sqsubseteq ss_1$. Also, $q'_1 \xrightarrow{\sigma,\Gamma} q'_2$ by the rule aut_2.

3. Inductive case: $a[\overrightarrow{T}] = (\star, n, b)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}]) = \mu(b[\overrightarrow{T}])$. By cases on inference rules

for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

- Case $\star_{1a}$: $s_1 = (\star_\circ, \mathit{started?}, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $s_1.\mathit{val} = s_2.\mathit{val} = \overrightarrow{V}$, $\mathit{final}(ss_1) \vee \neg\mathit{started?}$, $q \xrightarrow{\sigma,\Gamma} ss_2$ and $q \in \mathit{Abinit}(b[\overrightarrow{V}])$. Let $cc = \mu(b[\overrightarrow{T}]) = c$.
  - If $\mathit{started?}$, then we have $\mathit{final}(ss_1)$. By Lemma 21, take $qq_1 \in \mathcal{Q}^\sharp$ such that $\mathit{final}(qq_1)$, $\gamma(qq_1) \leq_k cc = c$ and $qq_1 \sqsubseteq ss_1$. By Lemma 13, for all $qq_1' \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq_1' \sqsubseteq ss_1$, $\mathit{final}(qq_1')$. By induction hypothesis, take $q', qq_2 \in \mathcal{Q}^\sharp$ such that $q' \xrightarrow{\sigma,\Gamma} qq_2$ with $\gamma(q') = \gamma(qq_2) \leq_k cc = c$, $q' \sqsubseteq q$, $qq_2 \sqsubseteq ss_2$ and for all $qq_2' \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$, there exists $q'' \in \mathcal{Q}^\sharp$ such that $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,\Gamma} qq_2'$. Let $\overrightarrow{W} = \mathit{sup}(qq_1.\mathit{val}, qq_2.\mathit{val}) \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k cc$. Take $qq_1' \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq_1' \sqsubseteq ss_1$ and $qq_1'.\mathit{val} = \overrightarrow{W}$, and $qq_2' \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$ and $qq_2'.\mathit{val} = \overrightarrow{W}$, which both exist by Lemma 19. We have $\mathit{final}(qq_1')$. And by induction hypothesis, take $q'' \in \mathcal{Q}^\sharp$ such that $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,\Gamma} qq_2'$. We have $q'' \in \mathit{Abinit}(b[\overrightarrow{W}])$ by Lemma 32. Let $q_1 = (\star_\circ, \mathit{started?}, qq_1')$ and $q_2 = (\star_\circ, \text{true}, qq_2')$. Thus, by the inference rule $\star_{1a}$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
    (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k cc = c$.
    (b) As $qq_1' \sqsubseteq ss_1$ and $qq_2' \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.
    (c) Let $q_2'' \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q_2'' \sqsubseteq s_2$ with $q_2''.\mathit{val} = \overrightarrow{V'}$. We have $q_2'' = (\star_\circ, \text{true}, qq_2'')$ with $qq_2' \sqsubseteq qq_2'' \sqsubseteq ss_2$. Take $q'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,\Gamma} qq_2'$. We have $q'' \in \mathit{Abinit}(b[\overrightarrow{V'}])$ by Lemma 32. Take $qq_1'' \in \mathcal{Q}^\sharp$ such that $qq_1''.\mathit{val} = q_2''.\mathit{val} = \overrightarrow{V'}$ and $qq_1' \sqsubseteq qq_1'' \sqsubseteq ss_1$, and which exists by Lemma 19. We have $\mathit{final}(qq_1'')$ by Lemma 13. Let $q_1'' = (\star_\circ, \mathit{started?}, qq_1'')$. We have $q_1 \sqsubseteq q_1'' \sqsubseteq s_1$. And by the rule $\star_{1a}$, $q_1'' \xrightarrow{\sigma,\Gamma} q_2''$.
  - If $\neg\mathit{started?}$, then take $qq_1 \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq ss_1$ and $\gamma(qq_1) \leq_k cc$, thanks to Lemma 20 instead of Lemma 21. The rest is similar to the case $\mathit{started?}$, without using the "final" conditions.
- Case $\star_2$: $s_1 = (\star_\circ, \text{true}, ss_1)$ and $s_2 = (\star_\circ, \text{true}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. Let $cc = \mu(b[\overrightarrow{T}])$. Similar to case aut_2.

4. Inductive case: $a[\overrightarrow{T}] = (|, n, l, r)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}]) = \mathit{sup}(\mu(l[\overrightarrow{T}]), \mu(r[\overrightarrow{T}]))$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

- Case $|_{1a}$: $s_1 = (|_\circ, \bot, \bot)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $s_1.\mathit{val} = s_2.\mathit{val} = \overrightarrow{V}$, $q \xrightarrow{\sigma,\Gamma} ss_2$ and $q \in \mathit{Abinit}(l[\overrightarrow{V}])$. Let $cc = \mu(ss_2.\mathit{pa}) = \mu(l[\overrightarrow{T}])$. By induction hypothesis, take $q', qq_2 \in \mathcal{Q}^\sharp$ such that $q' \xrightarrow{\sigma,\Gamma} qq_2$ with $\gamma(q') = \gamma(qq_2) \leq_k cc$, $q' \sqsubseteq q$, $qq_2 \sqsubseteq ss_2$ and for all $qq_2' \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$, there exists $q'' \in \mathcal{Q}^\sharp$ such that $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,\Gamma} qq_2'$. Let $\overrightarrow{W} = qq_2.\mathit{val}$. Let $q_1 = (|_\circ, \bot, \bot)$ and $q_2 = (|_\circ, \mathsf{left}, qq_2')$, with $q_1.\mathit{val} = q_2.\mathit{val} = \overrightarrow{W}$ and $q_1.\mathit{pa} = q_2.\mathit{pa} = a[\overrightarrow{T}]$. We have $q' \in \mathit{Abinit}(l[\overrightarrow{W}])$ by Lemma 32. Thus, by the inference rule $|_{1a}$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
  (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k cc \leq_k c$.

(b) As $\overrightarrow{W} \subseteq \overrightarrow{V}$ and $qq_2 \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

(c) Let $q'_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q'_2 \sqsubseteq s_2$ with $q'_2.val = \overrightarrow{V'}$. We have $q'_2 = (|_\circ, \mathsf{left}, qq'_2)$ with $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$. Take $q'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,\Gamma} qq'_2$. We have $q'' \in \mathit{Abinit}(l[\overrightarrow{V'}])$ by Lemma 32. Let $q'_1 = (|_\circ, \bot, \bot)$ with $q'_1.val = \overrightarrow{V'}$ and $q'_1.pa = a[\overrightarrow{T}]$. We have $q_1 \sqsubseteq q'_1 \sqsubseteq s_1$. And by the rule $|_{1a}$, $q'_1 \xrightarrow{\sigma,\Gamma} q'_2$.

- Case $|_{2a}$: same as previous with $r[\overrightarrow{T}]$.
- Case $|_3$: $s_1 = (|_\circ, \mathsf{left}, ss_1)$ and $s_2 = (|_\circ, \mathsf{left}, ss_2)$ with $ss_1 \xrightarrow{\sigma,\Gamma} ss_2$. Let $cc = \mu(ss_1.pa) = \mu(l[\overrightarrow{T}])$. Similar to the case aut_2.
- Case $|_4$: same as previous.

5. Inductive case: $a[\overrightarrow{T}] = (||, n, \Delta, l, r)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}]) = \mu(l[\overrightarrow{T}]) + \mu(r[\overrightarrow{T}])$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

   - Case $||_1$: $s_1 = (||_\circ, s_{l1}, s_{r1})$ and $s_2 = (||_\circ, s_{l2}, s_{r1})$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\alpha(\sigma) \notin \Delta$ and $s_{l1} \xrightarrow{\sigma,\Gamma} s_{l2}$. Let $cc_l = \mu(s_{l1}.pa) = \mu(l[\overrightarrow{T}])$ and $cc_r = \mu(s_{r1}.pa) = \mu(r[\overrightarrow{T}])$. By induction hypothesis, take $q_{l1}, q_{l2} \in \mathcal{Q}^\sharp$ such that $q_{l1} \xrightarrow{\sigma,\Gamma} q_{l2}$ with $\gamma(q_{l1}) = \gamma(q_{l2}) \leq_k cc_l$, $q_{l1} \sqsubseteq s_{l1}$, $q_{l2} \sqsubseteq s_{l2}$ and for all $q'_{l2} \in \mathcal{Q}^\sharp$ with $q_{l2} \sqsubseteq q'_{l2} \sqsubseteq s_{l2}$ there exists $q'_{l1} \in \mathcal{Q}^\sharp$ such that $q_{l1} \sqsubseteq q'_{l1} \sqsubseteq s_{l1}$ and $q'_{l1} \xrightarrow{\sigma,\Gamma} q'_{l2}$. Besides, by Lemma 20, take $q_{r1} \in \mathcal{Q}^\sharp$ such that $\gamma(q_{r1}) \leq_k cc_r$ and $q_{r1} \sqsubseteq s_{r1}$. Let $\overrightarrow{W} = q_{l1}.val \cup q_{r1}.val \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k \gamma(q_{l1}) + \gamma(q_{r1}) \leq_k cc_l + cc_r$. Take $q'_{l2}, q'_{r1} \in \mathcal{Q}^\sharp$ such that $q_{l2} \sqsubseteq q'_{l2} \sqsubseteq s_{l2}$, $q_{r1} \sqsubseteq q'_{r1} \sqsubseteq s_{r1}$ and $q'_{l2}.val = q'_{r1}.val = \overrightarrow{W}$ by Lemma 19. By the induction hypothesis, take $q'_{l1} \in \mathcal{Q}^\sharp$ such that $q_{l1} \sqsubseteq q'_{l1} \sqsubseteq s_{l1}$ and $q'_{l1} \xrightarrow{\sigma,\Gamma} q'_{l2}$. Let $q_1 = (||_\circ, q'_{l1}, q'_{r1})$ with $q_1.pa = a$ and $q_1.val = \overrightarrow{W}$ and $q_2 = (||_\circ, q'_{l2}, q'_{r1})$ with $q_2.pa = a$ and $q_2.val = \overrightarrow{W}$. By the rule $||_1$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.

     (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k cc_l + cc_r = c$.

     (b) As $q'_{l1} \sqsubseteq s_{l1}$, $q'_{l2} \sqsubseteq s_{l2}$ and $q'_{r1} \sqsubseteq s_{r1}$ then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

     (c) Let $q'_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q'_2 \sqsubseteq s_2$ with $q'_2.val = \overrightarrow{V'}$. We have $q'_2 = (||_\circ, q''_{l2}, q''_{r1})$ with $q'_2.pa = a$ and $q'_{l2} \sqsubseteq q''_{l2} \sqsubseteq s_{l2}$ and $q'_{r1} \sqsubseteq q''_{r1} \sqsubseteq s_{r1}$ and $q''_{l2}.val = q''_{r2} = \overrightarrow{V'}$. Take $q''_{l1} \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q'_{l1} \sqsubseteq q''_{l1} \sqsubseteq s_{l1}$ and $q''_{l1} \xrightarrow{\sigma,\Gamma} q''_{l2}$. Let $q'_1 = (||_\circ, q''_{l1}, q''_{r1})$ with $q'_1.pa = a$ and $q'_1.val = q''_{l1}.val = q''_{r1}.val = \overrightarrow{V'}$. We have $q_1 \sqsubseteq q'_1 \sqsubseteq s_1$ because $q'_{l1} \sqsubseteq q''_{l1} \sqsubseteq s_{l1}$ and $q'_{r1} \sqsubseteq q''_{r1} \sqsubseteq s_{r1}$. Finally, $q'_1 \xrightarrow{\sigma,\Gamma} q'_2$ by the rule $||_1$.

   - Case $||_2$: similar.
   - Case $||_3$: $s_1 = (||_\circ, s_{l1}, s_{r1})$ and $s_2 = (||_\circ, s_{l2}, s_{r1})$ with $s_1.val = s_2.val = \overrightarrow{V}$, $\alpha(\sigma) \in \Delta$, $s_{l1} \xrightarrow{\sigma,\Gamma} s_{l2}$ and $s_{r1} \xrightarrow{\sigma,\Gamma} s_{r2}$. Let $cc_l = \mu(s_{l1}.pa) = \mu(l[\overrightarrow{T}])$ and $cc_r = \mu(s_{r1}.pa) = \mu(r[\overrightarrow{T}])$. By induction hypothesis, take $q_{l1}, q_{l2} \in \mathcal{Q}^\sharp$ such that $q_{l1} \xrightarrow{\sigma,\Gamma} q_{l2}$ with $\gamma(q_{l1}) = \gamma(q_{l2}) \leq_k cc_l$, $q_{l1} \sqsubseteq s_{l1}$, $q_{l2} \sqsubseteq s_{l2}$ and for all $q'_{l2} \in \mathcal{Q}^\sharp$ with $q_{l2} \sqsubseteq q'_{l2} \sqsubseteq s_{l2}$ there exists $q'_{l1} \in \mathcal{Q}^\sharp$ such that $q_{l1} \sqsubseteq q'_{l1} \sqsubseteq s_{l1}$ and $q'_{l1} \xrightarrow{\sigma,\Gamma} q'_{l2}$. Similarly, take $q_{r1}, q_{r2} \in \mathcal{Q}^\sharp$ with

$q_{r1} \xrightarrow{\sigma,\Gamma} q_{r2}$, $\gamma(q_{r1}) = \gamma(q_{r2}) \leq_k cc_r$, $q_{r1} \sqsubseteq s_{r1}$, $q_{r2} \sqsubseteq s_{r2}$ ... Let $\overrightarrow{W} = q_{l1}.val \cup q_{r1}.val \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k cc_l + cc_r$. Take $q_{l2}', q_{r2}' \in \mathcal{Q}^\sharp$ such that $q_{l2} \sqsubseteq q_{l2}' \sqsubseteq s_{l2}$, $q_{r2} \sqsubseteq q_{r2}' \sqsubseteq s_{r2}$ and $q_{l2}'.val = q_{r2}'.val = \overrightarrow{W}$ by Lemma 19. By the induction hypothesis, take $q_{l1}', q_{r1}' \in \mathcal{Q}^\sharp$ such that $q_{l1} \sqsubseteq q_{l1}' \sqsubseteq s_{l1}$, $q_{r1} \sqsubseteq q_{r1}' \sqsubseteq s_{r1}$, $q_{l1}' \xrightarrow{\sigma,\Gamma} q_{l2}'$ and $q_{r1}' \xrightarrow{\sigma,\Gamma} q_{r2}'$. Let $q_1 = (||_\circ, q_{l1}', q_{r1}')$ with $q_1.pa = a$ and $q_1.val = \overrightarrow{W}$ and $q_2 = (||_\circ, q_{l2}', q_{r2}')$ with $q_2.pa = a$ and $q_2.val = \overrightarrow{W}$. By the rule $||_3$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.

(a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k cc_l + cc_r = c$.

(b) As $q_{l1}' \sqsubseteq s_{l1}$, $q_{l2}' \sqsubseteq s_{l2}$, $q_{r1}' \sqsubseteq s_{r1}$ and $q_{r2}' \sqsubseteq s_{r2}$ then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

(c) Let $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q_2' \sqsubseteq s_2$ with $q_2'.val = \overrightarrow{V'}$. We have $q_2' = (||_\circ, q_{l2}'', q_{r2}'')$ with $q_2'.pa = a$ and $q_{l2}' \sqsubseteq q_{l2}'' \sqsubseteq s_{l2}$ and $q_{r2}' \sqsubseteq q_{r2}'' \sqsubseteq s_{r2}$ and $q_{l2}''.val = q_{r2}'' = \overrightarrow{V'}$. Take $q_{l1}'', q_{r1}'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q_{l1}' \sqsubseteq q_{l1}'' \sqsubseteq s_{l1}$, $q_{r1}' \sqsubseteq q_{r1}'' \sqsubseteq s_{r1}$, $q_{l1}'' \xrightarrow{\sigma,\Gamma} q_{l2}''$ and $q_{r1}'' \xrightarrow{\sigma,\Gamma} q_{r2}''$. Let $q_1' = (||_\circ, q_{l1}'', q_{r1}'')$ with $q_1'.pa = a$ and $q_1'.val = q_{l1}''.val = q_{r1}''.val = \overrightarrow{V'}$. We have $q_1 \sqsubseteq q_1' \sqsubseteq s_1$ because $q_{l1}' \sqsubseteq q_{l1}'' \sqsubseteq s_{l1}$ and $q_{r1}' \sqsubseteq q_{r1}'' \sqsubseteq s_{r1}$. Finally, $q_1' \xrightarrow{\sigma,\Gamma} q_2'$ by the rule $||_3$.

6. Inductive case: $a[\overrightarrow{T}] = (|{:}, n, x, X, b)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}])$. By cases on inference rules for $s_1 \xrightarrow{\sigma,\Gamma} s_2$:

- Case $|{:}_1$: $s_1 = (|{:}_\circ, \bot, \bot)$ and $s_2 = (|{:}_\circ, v, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $q \xrightarrow{\sigma,([x:=v])\lhd\Gamma} ss_2$, $q \in Abinit(b[\overrightarrow{V}])$ and $v \in X$.
  - If X is the i-th parameter, then $c = \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0)$. Let $cc = \mu(ss_2.pa) = \mu(b[\overrightarrow{T}])$. By induction hypothesis, take $q', qq_2 \in \mathcal{Q}^\sharp$ such that $q' \xrightarrow{\sigma,([x:=v])\lhd\Gamma} qq_2$ with $\gamma(q') = \gamma(qq_2) \leq_k cc$, $q' \sqsubseteq q$, $qq_2 \sqsubseteq ss_2$ and for all $qq_2' \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$, there exists $q'' \in \mathcal{Q}^\sharp$ such that $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,([x:=v])\lhd\Gamma} qq_2'$. Let $\overrightarrow{W} = qq_2.val$ and $\overrightarrow{W'} = \overrightarrow{W} \cup (\emptyset, ..., \{v\}, ..., \emptyset)$ (the i-th component of $\overrightarrow{W}$ is augmented by v). We have $|\overrightarrow{W'}| \leq_k cc + (0, ..., 1, ..., 0)$. Let $qq_2' \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$ and $qq_2'.val = \overrightarrow{W'}$, which exists by Lemma 19. By induction hypothesis, take $q'' \in \mathcal{Q}^\sharp$ such that $q' \sqsubseteq q'' \sqsubseteq q$ and $q'' \xrightarrow{\sigma,([x:=v])\lhd\Gamma} qq_2'$. Let $q_1 = (|{:}_\circ, \bot, \bot)$ and $q_2 = (|{:}_\circ, v, qq_2')$, with $q_1.val = q_2.val = \overrightarrow{W'}$ and $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ (q_2 is valid as $v \in W_i'$). We have $q'' \in Abinit(b[\overrightarrow{W'}])$ by Lemma 32. Thus, by the inference rule $|{:}_1$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.

    (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W'}| \leq_k cc + (0, .., 1, .., 0) = c$.

    (b) As $\overrightarrow{W'} \subseteq \overrightarrow{V}$ and $qq_2' \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

    (c) Let $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q_2' \sqsubseteq s_2$ with $q_2'.val = \overrightarrow{V'}$. We have $q_2' = (|{:}_\circ, v, qq_2'')$ with $qq_2' \sqsubseteq qq_2'' \sqsubseteq ss_2$. Take $q''' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q'' \sqsubseteq q''' \sqsubseteq q$ and $q''' \xrightarrow{\sigma,([x:=v])\lhd\Gamma} qq_2''$. We have $q''' \in Abinit(b[\overrightarrow{V'}])$ by Lemma 32. Let $q_1' = (|{:}_\circ, \bot, \bot)$ with $q_1'.val = \overrightarrow{V'}$ and $q_1'.pa = a[\overrightarrow{T}]$. We have $q_1 \sqsubseteq q_1' \sqsubseteq s_1$. And by the rule $|{:}_1$, $q_1' \xrightarrow{\sigma,\Gamma} q_2'$.

    - If X is not a parameter, then the proof is similar except that $c = \mu(b[\overrightarrow{T}])$ and we do not augment $\overrightarrow{W}$ with v.

- Case $|:_2$: $s_1 = (|:_\circ, v, ss_1)$ and $s_2 = (|:_\circ, v, ss_2)$ with $s_1.val = s_2.val = \overrightarrow{V}$, $ss_1 \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} ss_2$ and $v \neq \bot$.
    - If X is the i-th parameter, then $c = \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0)$. Let $cc = \mu(ss_2.pa) = \mu(b[\overrightarrow{T}])$. By induction hypothesis, take $qq_1, qq_2 \in \mathcal{Q}^\sharp$ such that $qq_1 \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2$ with $\gamma(qq_1) = \gamma(qq_2) \leq_k cc$, $qq_1 \sqsubseteq ss_1$, $qq_2 \sqsubseteq ss_2$ and for all $qq_2' \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$ there exists $qq_1' \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq_1' \sqsubseteq ss_1$ and $qq_1' \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2'$. Let $\overrightarrow{W} = qq_1.val$ and $\overrightarrow{W'} = \overrightarrow{W} \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. We have $|\overrightarrow{W'}| \leq_k cc + (0, ..., 1, ..., 0)$. Let $qq_2' \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$ and $qq_2'.val = \overrightarrow{W'}$, which exists by Lemma 19. By induction hypothesis, take $qq_1' \in \mathcal{Q}^\sharp$ such that $qq_1 \sqsubseteq qq_1' \sqsubseteq ss_1$ and $qq_1' \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2'$. Let $q_1 = (|:_\circ, v, qq_1')$ with $q_1.pa = a$ and $q_1.val = \overrightarrow{W'}$ and $q_2 = (|:_\circ, v, qq_2')$ with $q_2.pa = a$ and $q_2.val = \overrightarrow{W'}$. By the rule $|:_2$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
        (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W'}| \leq_k cc + (0, ..., 1, ..., 0) = c$.
        (b) As $qq_1' \sqsubseteq ss_1$ and $qq_2' \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.
        (c) Let $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q_2' \sqsubseteq s_2$ with $q_2'.val = \overrightarrow{V'}$. We have $q_2' = (|:_\circ, v, qq_2'')$ with $q_2'.pa = a$ and $qq_2' \sqsubseteq qq_2'' \sqsubseteq ss_2$. Take $qq_1'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $qq_1' \sqsubseteq qq_1'' \sqsubseteq ss_1$ and $qq_1'' \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2''$. Let $q_1' = (|:_\circ, v, qq_1'')$ with $q_1'.pa = a$ and $q_1'.val = \overrightarrow{V'}$. We have $q_1 \sqsubseteq q_1' \sqsubseteq s_1$ because $qq_1' \sqsubseteq qq_1'' \sqsubseteq ss_1$. Finally, $q_1' \xrightarrow{\sigma,\Gamma} q_2'$ by the rule $|:_2$.
    - If X is not a parameter, then the proof is similar except that $c = \mu(b[\overrightarrow{T}])$ and we do not augment $\overrightarrow{W}$ with v.

7. Inductive case: $a[\overrightarrow{T}] = (|||:_:, n, x, X, b)[\overrightarrow{T}]$. Let $c = \mu(a[\overrightarrow{T}])$. By $|||:_1$, the only rule for $s_1 \xrightarrow{\sigma,\Gamma} s_2$, $s_1 = (|||:_\circ, f)$ and $s_2 = (|||:_\circ, f \Leftarrow \{v \mapsto ss_2\})$ with $s_1.val = s_2.val = \overrightarrow{V}$ and $f(v) \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} ss_2$.

- If X is the i-th parameter, then $c = \mu(b[\overrightarrow{T}]) + (0, ..., 1, ..., 0)$. Let $cc = \mu(ss_2.pa) = \mu(b[\overrightarrow{T}])$. By induction hypothesis, take $q, qq_2 \in \mathcal{Q}^\sharp$ such that $q \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2$ with $\gamma(q') = \gamma(qq_2) \leq_k cc$, $q \sqsubseteq f(v)$, $qq_2 \sqsubseteq ss_2$ and for all $qq_2' \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$, there exists $q' \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq q' \sqsubseteq f(v)$ and $q' \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2'$. Let $\overrightarrow{W} = qq_2.val$ and $\overrightarrow{W'} = \overrightarrow{W} \cup (\emptyset, ..., \{v\}, ..., \emptyset)$. We have $|\overrightarrow{W'}| \leq_k cc + (0, ..., 1, ..., 0)$. Let $qq_2' \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq_2' \sqsubseteq ss_2$ and $qq_2'.val = \overrightarrow{W'}$, which exists by Lemma 19. By induction hypothesis, take $q' \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq q' \sqsubseteq f(v)$ and $q' \xrightarrow{\sigma,([x:=v])\triangleleft\Gamma} qq_2'$. Let $q_1 = (|||:_\circ, \{v \mapsto q'\})$ and $q_2 = (|||:_\circ, \{v \mapsto qq_2'\})$, with $q_1.val = q_2.val = \overrightarrow{W'}$ and $q_1.pa = q_2.pa = a[\overrightarrow{T}]$. Thus, by the inference rule $|||:_1$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.
    (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W'}| \leq_k cc + (0, .., 1, .., 0) = c$.

(b) As $q' \sqsubseteq f(v)$ and $qq'_2 \sqsubseteq ss_2$, then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

(c) Let $q'_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q'_2 \sqsubseteq s_2$ with $q'_2.val = \overrightarrow{V'}$. We have $q'_2 = (|||{:}_\circ, f_2)$ with $f_2 = \{v \mapsto qq''_2, ...\}$ and $qq'_2 \sqsubseteq qq''_2 \sqsubseteq ss_2$. Take $q'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q' \sqsubseteq q'' \sqsubseteq f(v)$ and $q'' \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} qq''_2$. Let $q'_1 = (|||{:}_\circ, f_1)$ with $f_1 = f_2 \Leftarrow \{v \mapsto q''\}$, $q'_1.val = \overrightarrow{V'}$ and $q'_1.pa = a[\overrightarrow{T}]$. Trivially, $q_1 \sqsubseteq q'_1$. For all $v' \in dom(f_2)$ such that $v' \neq v$, $f_2(v') \sqsubseteq f(v')$ and $q'' \sqsubseteq f(v)$. Thus, for all $v' \in dom(f_1)$, $f_1(v') \sqsubseteq f(v')$. Consequently, $q'_1 \sqsubseteq s_1$. Finally, by the rule $|||{:}_1$, $q'_1 \xrightarrow{\sigma,\Gamma} q'_2$.

- If X is not a parameter, then $c = |X| \cdot \mu(b[\overrightarrow{T}])$. Let $cc = \mu(ss_2.pa) = \mu(b[\overrightarrow{T}])$. By induction hypothesis, take $q, qq_2 \in \mathcal{Q}^\sharp$ such that $q \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} qq_2$ with $\gamma(q') = \gamma(qq_2) \leq_k cc$, $q \sqsubseteq f(v)$, $qq_2 \sqsubseteq ss_2$ and for all $qq'_2 \in \mathcal{Q}^\sharp$ with $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$, there exists $q' \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq q' \sqsubseteq f(v)$ and $q' \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} qq'_2$. Besides, by Lemma 20, for all $ss_{1,i} \in ran(f)$ such that $ss_{1,i} \neq ss_2$, take $qq_{1,i} \in \mathcal{Q}^\sharp$ such that $\gamma(qq_{1,i}) \leq_k cc$ and $qq_{1,i} \sqsubseteq ss_{1,i}$. Let $\overrightarrow{W} = \bigcup_i qq_{1,i}.val \cup qq_2.val \subseteq \overrightarrow{V}$. We have $|\overrightarrow{W}| \leq_k \sum_i \gamma(qq_{1,i}) + \gamma(qq_2) \leq_k |X| \cdot cc$. Take $qq'_2 \in \mathcal{Q}^\sharp$ such that $qq_2 \sqsubseteq qq'_2 \sqsubseteq ss_2$ and $qq'_2.val = \overrightarrow{W}$ by Lemma 19. Likewise, for all $qq_{1,i}$, take $qq'_{1,i} \in \mathcal{Q}^\sharp$ such that $qq_{1,i} \sqsubseteq qq'_{1,i} \sqsubseteq ss_{1,i}$ and $qq'_{1,i}.val = \overrightarrow{W}$. By the induction hypothesis, take $q' \in \mathcal{Q}^\sharp$ such that $q \sqsubseteq q' \sqsubseteq f(v)$ and $q' \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} qq'_2$. Let $q_1 = (|||{:}_\circ, f_1)$ and $q_2 = (|||{:}_\circ, f_2)$, with $q_1.val = q_2.val = \overrightarrow{W}$, $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and where f_1, f_2 are such that $dom(f_1) = dom(f_2) = dom(f)$, $f_1(f^{-1}(ss_{1,i})) = f_2(f^{-1}(ss_{1,i})) = qq'_{1,i}$, $f_1(v) = q'$ and $f_2(v) = qq'_2$. Thus, by the inference rule $|||{:}_1$, we have $q_1 \xrightarrow{\sigma,\Gamma} q_2$.

  (a) We have $\gamma(q_1) = \gamma(q_2) = |\overrightarrow{W}| \leq_k |X| \cdot cc = c$.

  (b) As $qq'_1 \sqsubseteq ss_1$, $qq'_2 \sqsubseteq ss_2$ and for all i $qq'_{1,i} \sqsubseteq ss_{1,i}$ then $q_1 \sqsubseteq s_1$ and $q_2 \sqsubseteq s_2$.

  (c) Let $q'_2 \in \mathcal{Q}^\sharp$ such that $q_2 \sqsubseteq q'_2 \sqsubseteq s_2$ with $q'_2.val = \overrightarrow{V'}$. We have $q'_2 = (|||{:}_\circ, f'_2)$ with $f'_2 = \{v \mapsto qq''_2, ...\}$ and $qq'_2 \sqsubseteq qq''_2 \sqsubseteq ss_2$. Take $q'' \in \mathcal{Q}^\sharp$ satisfying the induction hypothesis, *i.e.* $q' \sqsubseteq q'' \sqsubseteq f(v)$ and $q'' \xrightarrow{\sigma,(\![x:=v]\!)\lhd\Gamma} qq''_2$. Let $q'_1 = (|||{:}_\circ, f'_1)$ with $f'_1 = f'_2 \Leftarrow \{v \mapsto q''\}$, $q'_1.val = \overrightarrow{V'}$ and $q'_1.pa = a[\overrightarrow{T}]$. For all $v' \in dom(f)$ such that $v' \neq v$, $f_1(v') = f_2(v') \sqsubseteq f'_2(v') = f'_1(v') \sqsubseteq f(v')$ and $q' \sqsubseteq q'' \sqsubseteq f(v)$. Thus, for all $v' \in dom(f)$, $f_1(v') \sqsubseteq f'_1(v') \sqsubseteq f(v')$. Consequently, $q_1 \sqsubseteq q'_1 \sqsubseteq s_1$. Finally, by the rule $|||{:}_1$, $q'_1 \xrightarrow{\sigma,\Gamma} q'_2$.

□

Lemma 22. Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. For all $s_1, s_2 \in \mathcal{Q}^\sharp$ such that $s_1.pa = s_2.pa = a[\overrightarrow{T}]$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$, there exist $q_1, q_2 \in \mathcal{Q}^\sharp$ such that $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and $q_1 \xrightarrow{\sigma,\Gamma} q_2$ and:

1. $\gamma(q_1) = \gamma(q_2) \leq_k c$, and

2. $q_1 \preceq s_1$ and $q_2 \preceq s_2$, and

3. for all $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \preceq q_2' \preceq s_2$, there exists $q_1' \in \mathcal{Q}^\sharp$ such that $q_1 \preceq q_1' \preceq s_1$ and $q_1' \xrightarrow{\sigma',\Gamma'} q_2'$, where σ' and Γ' are the renamed event and environment with regards to the rename function ξ such that $q_2 \sqsubseteq \xi(q_2')$.

Proof. Let $a[\overrightarrow{T}]$ be a PASTD. Let $c \in \mathbb{N}^k$ such that $c = \mu(a[\overrightarrow{T}])$. Let $s_1, s_2 \in \mathcal{Q}^\sharp$ such that $s_1.pa = s_2.pa = a[\overrightarrow{T}]$ and $s_1 \xrightarrow{\sigma,\Gamma} s_2$. Take $q_1, q_2 \in \mathcal{Q}^\sharp$ verifying Lemma 33. We have $q_1.pa = q_2.pa = a[\overrightarrow{T}]$ and $q_1 \xrightarrow{\sigma,\Gamma} q_2$.

1. We have $\gamma(q_1) = \gamma(q_2) \leq_k c$.

2. We have $q_1 \sqsubseteq s_1 \implies q_1 \preceq s_1$ and $q_2 \sqsubseteq s_2 \implies q_2 \preceq s_2$.

3. Let $q_2' \in \mathcal{Q}^\sharp$ such that $q_2 \preceq q_2' \preceq s_2$. Let $q_2'' = \xi(q_2')$ such that $q_2 \sqsubseteq q_2'' \sqsubseteq s_2$ by Definition 35, with ξ an adequate rename function (it exists because $q_2.val \subseteq s_2.val$). Then, by Lemma 33, take $q_1'' \in \mathcal{Q}^\sharp$ such that $q_1 \sqsubseteq q_1'' \sqsubseteq s_1$ and $q_1'' \xrightarrow{\sigma,\Gamma} q_2''$. Let $q_1' \in \mathcal{Q}^\sharp$ such that $q_1'' = \xi(q_1')$. We have $q_1 \sqsubseteq q_1'' \sqsubseteq s_1$. And by Lemma 17, $q_1' \xrightarrow{\sigma',\Gamma'} q_2'$, where σ' and Γ' are the renamed event and environment with regards to the rename function ξ^{-1}.

□

Bibliography

[1] P. A. Abdulla, K. Cerans, B. Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 313–321, Jul 1996.

[2] Parosh Aziz Abdulla, Frédéric Haziza, and Lukáš Holík. *All for the Price of Few*, pages 476–495. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] Krzysztof R. Apt and Dexter C. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22(6):307 – 309, 1986.

[4] J.A. Bergstra and J.W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1):109 – 137, 1984.

[5] P. Bernus, G. Schmidt, and K. Mertins, editors. *Handbook on Architectures of Information Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[6] Jesse D. Bingham and Alan J. Hu. Empirically efficient verification for a class of infinite-state systems. In *Tools and Algorithms for the Construction and Analysis of Systems, 11th International Conference (TACAS 2005)*, pages 77–92, 2005.

[7] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Comput. Netw. ISDN Syst.*, 14(1):25–59, March 1987.

[8] Raphaël Chane-Yack-Fa. *Vérification formelle de systèmes d'information*. PhD thesis, Université de Sherbrooke, 2017.

[9] Raphaël Chane-Yack-Fa. Verification of Parameterized Algebraic State Transition Diagrams. Technical report, Département d'informatique, Faculté des Sciences, Université de Sherbrooke, 2017. `http://www.dmi.usherb.ca/~frappier/Papers/pastd.pdf`.

[10] Edmund Clarke, Muralidhar Talupur, and Helmut Veith. *Environment Abstraction for Parameterized Verification*, pages 126–141. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[11] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, pages 313–327, 2010.

[12] Leonard Eugene Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):pp. 413–422, 1913.

[13] Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992.

[14] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98 Aalborg, Denmark, July 13–17, 1998 Proceedings*, pages 103–115. Springer Berlin Heidelberg, 1998.

[15] Michel Embe-Jiague, Marc Frappier, Frederic Gervais, Pierre Konopacki, Regine Laleau, Jeremy Milhau, and Richard St-Denis. Model-driven engineering of functional security policies. In *12th International Conference on Enterprise Information Systems (ICEIS)*, pages 374–379, 2010.

[16] E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In *Automated Deduction - CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000, Proceedings*, pages 236–254, 2000.

[17] E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1-2):105–131, August 1996.

[18] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63 – 92, 2001. ISS.

[19] Alain Finkel. *A generalization of the procedure of karp and miller to well structured transition systems*, pages 499–508. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[20] Alain Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3):129–135, 1994.

[21] M. Frappier and R. St-Denis. EB^3: an entity-based black-box specification method for information systems. *Software and Systems Modeling*, 2(2):134–149, 2003.

[22] Marc Frappier, Benoît Fraikin, Romain Chossart, Raphaël Chane-Yack-Fa, and Mohammed Ouenzar. Comparison of model checking tools for information systems. In *Formal Methods and Software Engineering - 12th International Conference on Formal Engineering Methods, ICFEM 2010, Shanghai, China, November 17-19, 2010. Proceedings*, pages 581–596, 2010.

[23] Marc Frappier, Frédéric Gervais, Régine Laleau, and Benoît Fraikin. Algebraic State Transition Diagrams. Technical report, Université de Sherbrooke, 2008. `http://www.dmi.usherb.ca/~frappier/Papers/astd.pdf`.

[24] Marc Frappier, Frédéric Gervais, Régine Laleau, Benoît Fraikin, and Richard St-Denis. Extending statecharts with process algebra operators. *ISSE*, 4(3):285–292, 2008.

[25] Youssef Hanna, David Samuelson, Samik Basu, and Hridesh Rajan. Automating cut-off for multi-parameterized systems. In *Formal Methods and Software Engineering - 12th International Conference on Formal Engineering Methods, ICFEM 2010, Shanghai, China, November 17-19, 2010. Proceedings*, pages 338–354, 2010.

[26] Graham Higman. Ordering by Divisibility in Abstract Algebras. *Proceedings of the London Mathematical Society*, pages 326–336, 1952.

[27] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, August 1978.

[28] John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135 – 159, 1979.

[29] Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, pages 645–659, 2010.

[30] Barbara König and Jan Stückrath. *A General Framework for Well-Structured Graph Transformation Systems*, pages 467–481. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[31] Joseph B Kruskal. Well-quasi-ordering, the Tree Theorem, and Vazsonyi's Conjecture. *Transactions of the American Mathematical Society*, pages 210–225, 1960.

[32] Kenneth L. McMillan. Verification of infinite state systems by compositional model checking. In *Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, CHARME '99, pages 219–234, 1999.

[33] Roland Meyer. *On Boundedness in Depth in the π-Calculus*, pages 477–489. Springer US, Boston, MA, 2008.

[34] R. Milner. *Communication and Concurrency.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[35] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[36] Neil Robertson and Paul D. Seymour. Graph minors XXIII. Nash-Williams' immersion conjecture. *J. Comb. Theory, Ser. B*, 100(2):181–205, 2010.

[37] A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[38] Sylvain Schmitz and Philippe Schnoebelen. Algorithmic Aspects of WQO Theory. Lecture Notes, August 2012.

[39] Antti Siirtola and Juha Kortelainen. Algorithmic verification with multiple and nested parameters. In *Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings*, pages 561–580, 2009.

[40] Antti Siirtola and Juha Kortelainen. Parameterised process algebraic verification by precongruence reduction. In *Ninth International Conference on Application of Concurrency to System Design, ACSD 2009, Augsburg, Germany, 1-3 July 2009*, pages 158–167, 2009.

[41] Dimitris Vekris, Frédéric Lang, Catalin Dima, and Radu Mateescu. Verification of EB^3 specifications using CADP. In *Integrated Formal Methods, 10th International Conference, IFM 2013, Turku, Finland, June 10-14, 2013. Proceedings*, pages 61–76, 2013.