Parameterized verification of monotone information systems

Raphael Chane-Yack-Fa, Marc Frappier, Amel Mammar, Alain Finkel

Parameterized Verification of Monotone Information Systems

Raphaël Chane-Yack-Fa[1], Marc Frappier[1], Amel Mammar[2] and Alain Finkel[3]

[1]GRIL, Département d'informatique, Faculté des sciences, Université de Sherbrooke, Canada
[2]Télécom SudParis, SAMOVAR-CNRS, Évry, France
[3]LSV, CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

Abstract. In this paper, we study the information system verification problem as a parameterized verification one. Informations systems are modeled as multi-parameterized systems in a formal language based on the Algebraic State-Transition Diagrams (ASTD) notation. Then, we use the Well Structured Transition Systems (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, *i.e.* the computability of a base of predecessors for every states.

Keywords: model checking; parameterized verification; process algebra; well-structured transition systems; well-quasi-ordering; coverability; information systems

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 [BSM99]. 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 [McM99, EK00, CTV06, Mey09, SK09b, SK09a, HSBR10, KKW10, AHH13, DSZ10, KS14] 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

Correspondence and offprint requests to: Marc Frappier, Département d'informatique, Faculté des sciences, Université de Sherbrooke, Canada. e-mail: Marc.Frappier@USherbrooke.ca

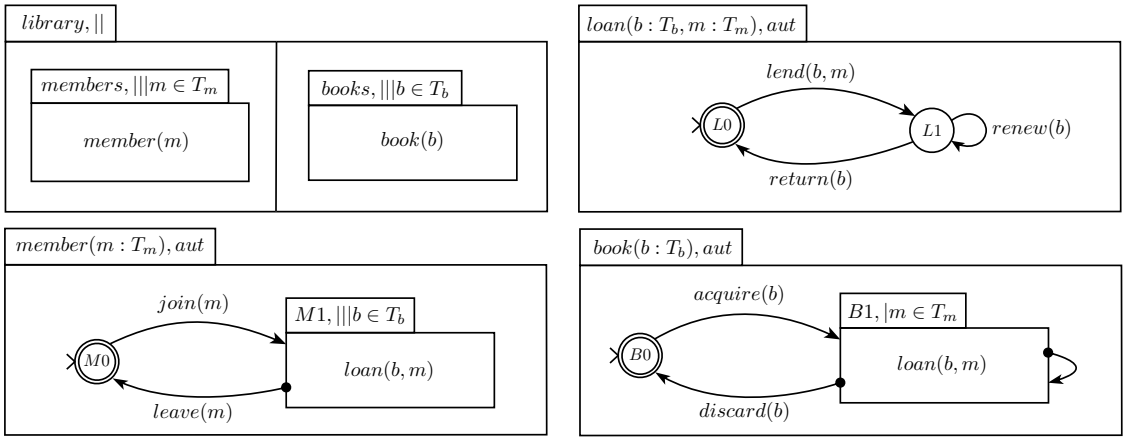

Fig. 1. Example of a library system

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 adapted fragment of the *Algebraic State-Transition Diagram* (ASTD) notation, proposed in [FGL+08], 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 [FS01] 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 instances must be able to fire a similar transition. Moreover, to ensure termination, we devise some structural conditions on the states. However, unlike [FS01, DSZ10, Mey09, KS14], we deal with those conditions separately as we describe a general semi-decision procedure. 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 2 presents the ASTD notation and the PASTD fragment. In Section 3, we recall the WSTS theory and show how to apply it to PASTDs by determining adequate conditions. Section 4 presents the RMTS framework and proves that PASTDs are RMTSs. The related work is presented in Section 5 and Section 6 concludes.

2. A Visual Process Algebra

An *Algebraic State-Transition Diagram* (ASTD) [FGL+08] is a graphical notation combining automata, statecharts [Har87] and process algebras to describe complex dynamic systems like information systems. ASTDs are closely related to process algebras like CSP [Hoa78], CCS [Mil89], ACP [BK84], LOTOS [BB87] and EB^3 [FSD03]. 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 [RHB97]. 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 [FGL+08]. For example, they make explicit the handling of instances of information system entities by using quantifications. Furthermore, the model has been used in [EJFG+10] to specify access control and security policies.

For instance, an ASTD model of a library system is given in Figure 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 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.

2.1. Parameterized ASTD

In this paper, we restrict ourselves to a fragment of the ASTD language. The graphical and textual notation of the original ASTDs are detailed in [FGLF08].
Let us denote a labeled tree by an expression thanks to the grammar $Tree ::= Node \mid Node[Tree, \ldots, Tree]$.

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

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

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; the transition relation δ is given by a set of tuples $(q_1, q_2, \sigma, final?)$, where $q_1, q_2 \in Q$, $\sigma \in \Sigma$ is an event and $final?$ is a boolean denoting a transition that can be fired only from a final state (represented by a big dot at the origin of the arrow); an event is noted $l(v_1, \ldots, v_n)$ where l is the event label, and v_i are event parameters; function α extracts the label of an event: $\alpha(l(v_1, \ldots, v_n)) = l$;
- $\mathbf{F} \,||_{\Delta}\, \mathbf{F}$ 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 $|_{x \in T} \mathbf{F}$ allows us to pick a value v from a finite set T and execute its component ASTD $\mathbf{F}$, where every occurrence of the variable x is replaced by the value v;
- a Quantified Interleaving ASTD $|||_{x \in T} \mathbf{F}$ allows us to execute as many interleaving instances of $\mathbf{F}$ 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 introduce the following tree-like notation, which is easier to manipulate:

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

$$
\begin{array}{ll}
\mathbf{F'} ::= \mathbf{F} & \\
\quad | \quad \epsilon &
\end{array}
$$

Note that ϵ is used to create elementary automaton states. Figure 2 shows the tree notation of the ASTD of Figure 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 if one of the sub-processes is in a final state, *i.e.* there is no final synchronization between all interleaving processes (see Definition 23 of Appendix A). Indeed, the property of monotony, that we will explain in Section 3.2, is easier to obtain without quantified synchronizations.
The quantified operators (interleaving and choice) allow us to model replication of processes and interactions

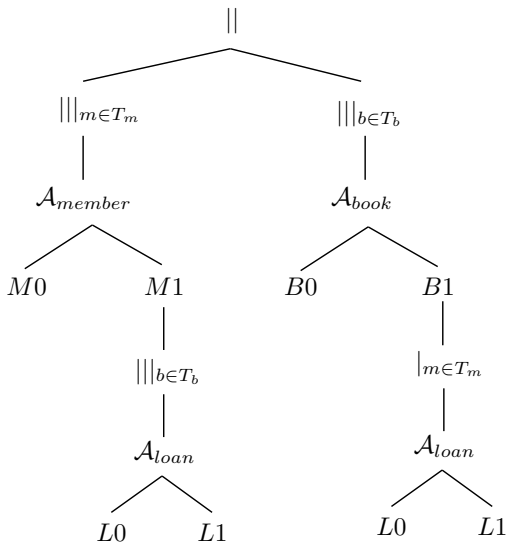

Fig. 2. The tree representation of an ASTD

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 and 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, $\overrightarrow{T} = (T_1, \ldots, T_n)$ is a vector of n variables, called parameters, and $\overrightarrow{P} = (P_1, \ldots, P_n)$ 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 $U_1 \subseteq P_1, \ldots, U_n \subseteq P_n$, the substitution of $\overrightarrow{T}$ by $\overrightarrow{U}$ 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 $P_m = \{m_1, m_2 \ldots\}$ and $P_b = \{b_1, b_2 \ldots\}$, 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 is given by a tree structure.

Definition 3 (ASTD state). A PASTD state is always associated with an ASTD expression with which it must be consistent. ASTD states are defined inductively by the following grammar:

$\mathbf{S}$ ::=	$(\mathsf{aut}_\circ, q)$	(automaton, elementary state q)
	$(\mathsf{aut}_\circ, q)[\mathbf{S}]$	(automaton, composite state q)
	$\|\|_\circ[\mathbf{S}, \mathbf{S}]$	(synchronization)
	$\|:_\circ$	(quantified choice, value not chosen yet)
	$\|:_\circ[p[\mathbf{S}]]$	(quantified choice, p is chosen)
	$\|\|\|:_\circ[p_1[\mathbf{S}], \ldots, p_k[\mathbf{S}]]$	(quantified interleaving on $\{p_1, \ldots, p_k\}$, $k \in \mathbb{N}_1$)

Let s be a state of PASTD A. For s to be consistent with A, the following constraints must be satisfied:

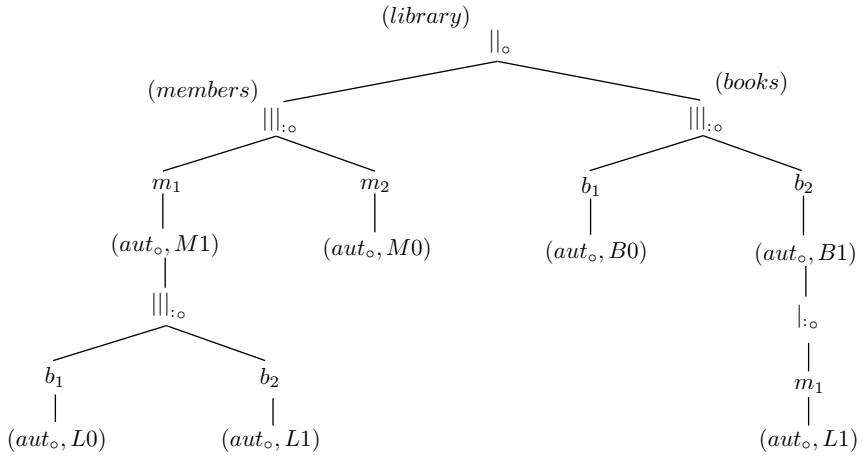

Fig. 3. A state of the library system

$s = (\mathsf{aut}_\circ, q)$	:	$A = (Q, \Sigma, \delta, q_0, Q_f) \wedge q \in Q$ and q is elementary
$s = (\mathsf{aut}_\circ, q)[s']$	:	$A = (Q, \Sigma, \delta, q_0, Q_f) \wedge q \in Q$ and s' is consistent with the PASTD of composite state q
$s = \|\|_\circ[s', s'']$	:	$A = A' \,\|\|_\Delta\, A''$ and s' is consistent with A' and s'' is consistent with A''
$s = \|:_\circ$	:	$A = \|_{x \in T} A'$
$s = \|:_\circ[p[s']]$	:	$A = \|_{x \in T} A' \wedge p \in T$ and s' is consistent with A'
$s = \|\|\|:_\circ[\ldots, p[s'], \ldots]$	:	$A = \|\|\|_{x \in T} A' \wedge p \in T$ and s' is consistent with A'

For a PASTD A, we denote by $\mathcal{T}_A$ the set of all well-formed and consistent 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 an undirected 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. A labeled graph is 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 $Tree ::= Node \mid Node[Tree, \ldots, Tree]$ 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. For a PASTD $A = (F, (T_1, \ldots, T_n), (P_1, \ldots, P_n))$ and a state $s = (V, E, \lambda) \in \mathcal{T}_A$, we denote by $val(s) = (R_1, \ldots, R_n)$ 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. 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 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 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 $\to$ and we obtain the transition system $\mathcal{S}_A = (\mathcal{T}_A, \to)$. Intuitively, $||_\circ$ makes a transition if both its sub-branches are able to do the transition on the same event. $|||_{:\circ}$ can do the transition if one of its branches can and $|_{:\circ}$ if its sub-branch can. Finally, an automaton state fires a transition either by a classical automaton transition or within a sub-state.

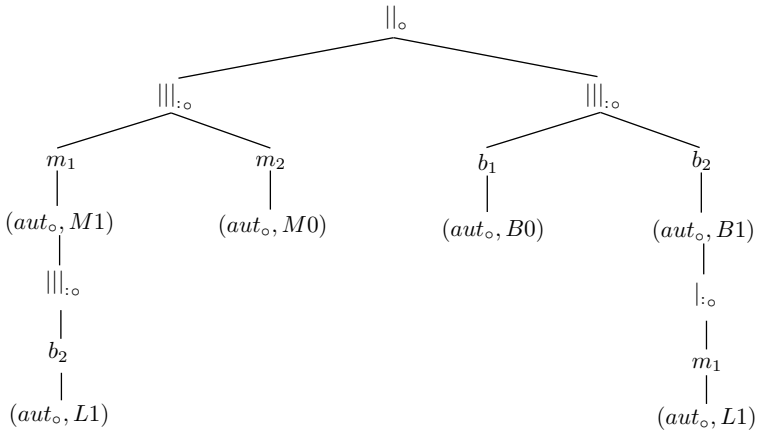

Fig. 4. An abstract state

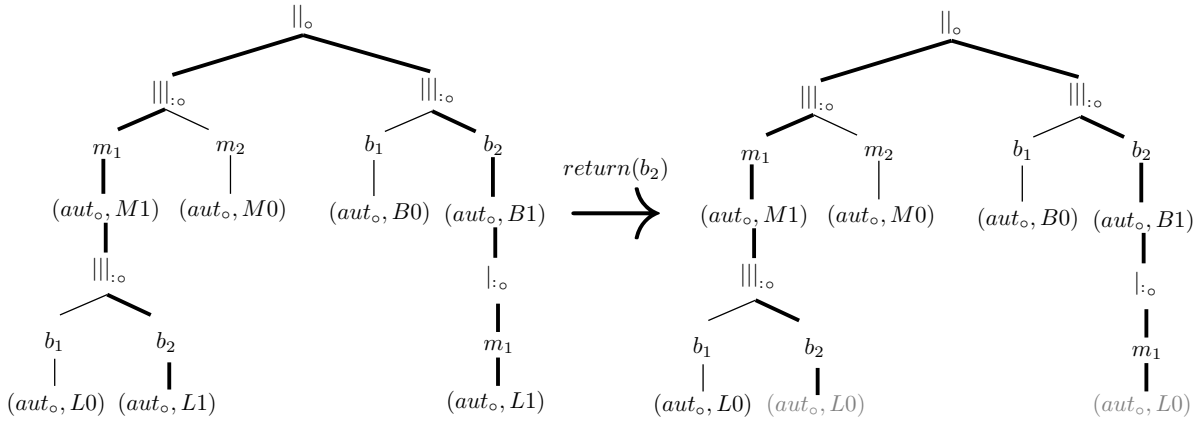

Fig. 5. A transition

Figure 5 shows an example of transition in the library system where the member m_1 returns the book b_2. The branches involved in the transition are depicted by bold strokes and the notable change in red. For a complete description of the operational semantics of ASTDs and PASTDs, see Appendix A or [FGLF08]. Note that the transition system of a PASTD $A = (F, (T_1, \ldots, T_n), (P_1, \ldots, P_n))$ correspond to the infinite union of transition systems for ASTDs $F[(T_1, \ldots, T_n) := (R_1, \ldots, R_n)]$ for all $(R_1, \ldots, R_n) \subset (P_1, \ldots, P_n)$ (see [CYF17]).

2.2. Expressiveness of PASTD

Considering the previous extension of ASTDs with parameters, we can show that PASTDs are more expressive than some types of infinite systems like *Petri Nets* [Pet81] or *Vector Addition Systems with States* (VASS) [HP79]. 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 $(p, (u_1, \ldots, u_n)) \to (p', (u'_1, \ldots, u'_n))$ if $(p, (a_1, \ldots, a_n), p') \in E$, 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 $q \to q_1 \to q_2 \to \cdots \to q'$, called a path. We say that a transition system

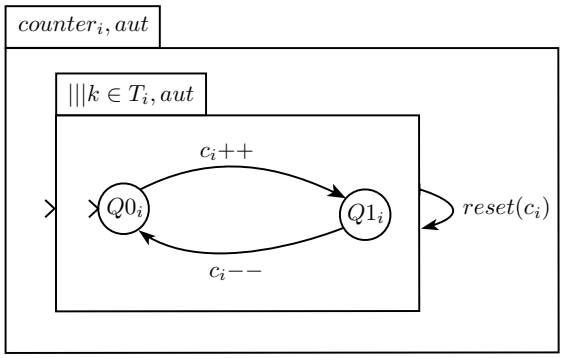

Fig. 6. ASTD model of a counter

$(Q, \rightarrow)$ simulates another transition system $(Q', \hookrightarrow)$ if there exists an injection $f : Q' \rightarrow Q$ 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 $\{c_1, \ldots, c_n\}$ and of a graph called control graph. Each arc of the graph is labeled by an action that can change the values of the counters.

Proposition 1. PASTDs simulate RVASSs.

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 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 $Q1_i$. 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 increase or decrease only one counter at the same time and only by one. Thus, we need to split some transitions into sequences of transitions by adding new control states.
- Finally, we put together all counters thanks to interleaving PASTDs. Then, the global PASTD modeling the RVASS is represented by a synchronization PASTD between the control graph and the counters. Each time a counter c_i is incremented by a $c_i{+}{+}$ action, one instance of automaton in the quantified interleaving is non-deterministically chosen to move from $Q0_i$ to $Q1_i$. Similarly, a $c_i{-}{-}$ action decrements the counter c_i only if there is an automaton in $Q1_i$ to move to $Q0_i$. A $reset(c_i)$ action will reinitialize all automaton to their initial states, *i.e.* $Q0_i$. Note that instances in quantified interleaving are not generated dynamically during transitions, but each sequence of states exists in the infinite union of instantiated systems. Thus, this construction works for unbounded counters.

A configuration of RVASS is represented by a set of states in the PASTD, which happens to be an equivalence class (as shown further in Section 3.3). Each transition of configuration of the RVASS is mimicked by a transition or a sequence of transitions in the resulting PASTD. A transition of states in the PASTD corresponds to a transition of equivalence classes (as will be shown in Section 4.2). As a consequence, PASTDs can simulate RVASSs. See an example of simulation in Figure 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 [DFS98]. Since simulation preserves reachability of states, we can conclude that the reachability problem of equivalence classes is undecidable for PASTDs (see Section 4.2).

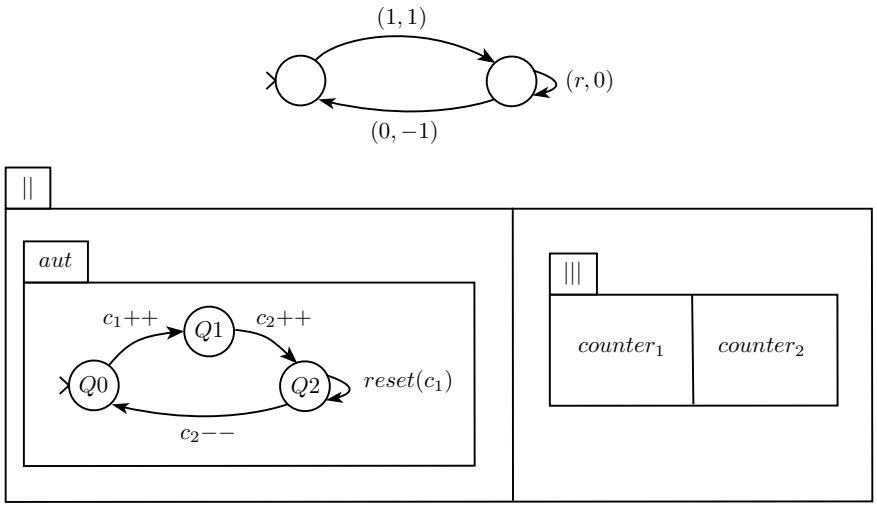

Fig. 7. An RVASS and its PASTD simulation

3. Well-Structured PASTD

3.1. Preliminaries

For a transition system $S = (Q, \to)$, we define $Pred(q) = \{q' \in Q \mid q' \to q\}$ 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 2 ([Hig52]). 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 $x_{n_0} \leq x_{n_1} \leq \cdots$ with $n_0 < n_1 < \cdots$;
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 $U_0 \subseteq U_1 \subseteq \cdots \subseteq U_i \subseteq \cdots$ of upward-closed subsets U_i of X eventually stabilizes.

3.2. Well-Structured Transition Systems

The theory of *Well-Structured Transition Systems* [Fin87, ACJT96, FS01] 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, because of the monotony condition, 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 [ACJT96], 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 $Pred(\uparrow s)$, $Pred^2(\uparrow s) = Pred(Pred(\uparrow s)), \ldots, Pred^n(\uparrow s) \ldots$ and we have $Pred^*(\uparrow s) = \bigcup_{n \in \mathbb{N}} Pred^n(\uparrow s)$. 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 $Pred^*(\uparrow s)$ 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 [FS01]). 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 [FS01], 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 [FS01]). 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 transition system is monotone. Lossy Channel Systems [Fin94] 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 ([FS01]). 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 [FS01]). 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 $\uparrow Pred(\uparrow q)$ (or pred-basis).

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 $K_0, K_1 \ldots$ such that $K_0 = F$ and $K_{n+1} = K_n \cup pb(K_n)$. We may verify that $Pred^*(\uparrow F) = \bigcup_{i \in \mathbb{N}} \uparrow K_i$.

Lemma 2 ([FS01]). 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 ([FS01]). 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.

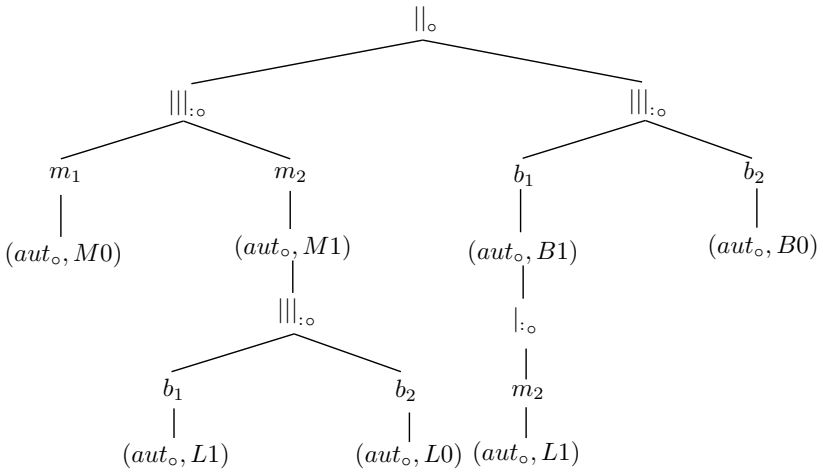

Fig. 8. A state of the library system

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 [FS01].

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 $f : X \to X$. A *graph isomorphism* from (V, E) to (V', E') is a bijective function $f : V \to V'$ 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 where 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 $\{(\mathsf{aut}_\circ, q_1), \ldots, (\mathsf{aut}_\circ, q_j), ||_\circ, |:_\circ, |||:_\circ\} \cup \bigcup_i P_i$. Definition 8 means that 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 if all occurrences are renamed similarly. For example, the state of Figure 3 is equivalent to the state of Figure 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 $f : X \to Y$ by $g : W \to Y$, *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 an *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).

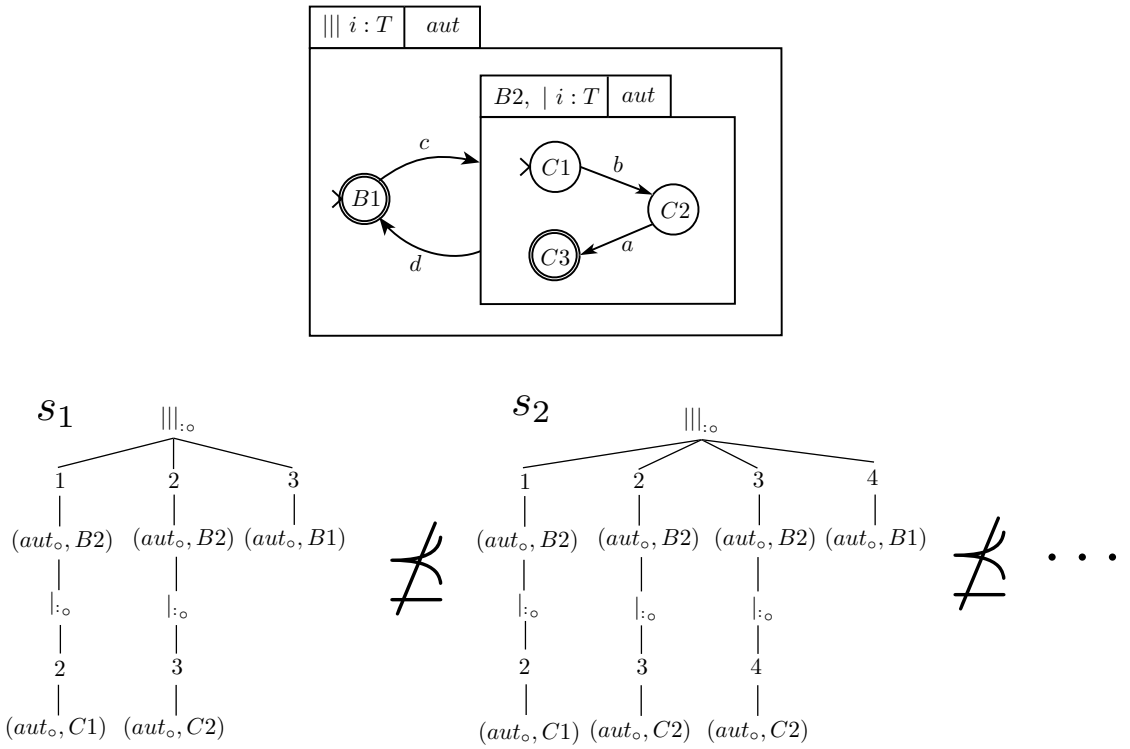

Fig. 9. An infinite antichain in $(\mathcal{T}_A, \preceq)$

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

Intuitively, we have $s \preceq s'$ if s is a subtree of s' modulo renaming. Furthermore, 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 3. PASTDs under $\preceq$ are monotone transition systems.

Proof. The proof of monotony is similar to the one given in [CYF17] for a superset of PASTD. □

3.4. Bounded-PASTD

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

Proposition 4. There is a PASTD such that $\preceq$ is not wqo.

Proof. Figure 9 shows an example of infinite antichain for a PASTD. We construct the antichain similarly to the infinite antichain $\{abc, abbc, abbbc \ldots\}$ for the prefix ordering, or the H graphs for the subgraph ordering [Din92]. Consider the state s_1 with 3 branches of interleaving, where $val(s_1) = (\{1, 2, 3\})$. The first branch of the interleaving is linked to the second one because they share the value 2. Likewise, the second branch is linked to the last one by the value 3. We construct s_2 with $val(s_2) = (\{1, 2, 3, 4\})$ by adding a new branch between the second and the last branch of s_1 and removing the direct link between them. Thus, we have $s_1 \not\preceq s_2$. We construct the other states of the infinite antichain following the same principle. By Proposition 2, $\preceq$ is not wqo. □

We have not found yet a useful wqo for all PASTDs. Nonetheless, for some PASTDs, the set of states is wqo. For example, it is obviously the case for PASTDs without parameters, as the set of states is then finite. Some other weaker conditions can be used to get a wqo. We define in the following a class of PASTDs

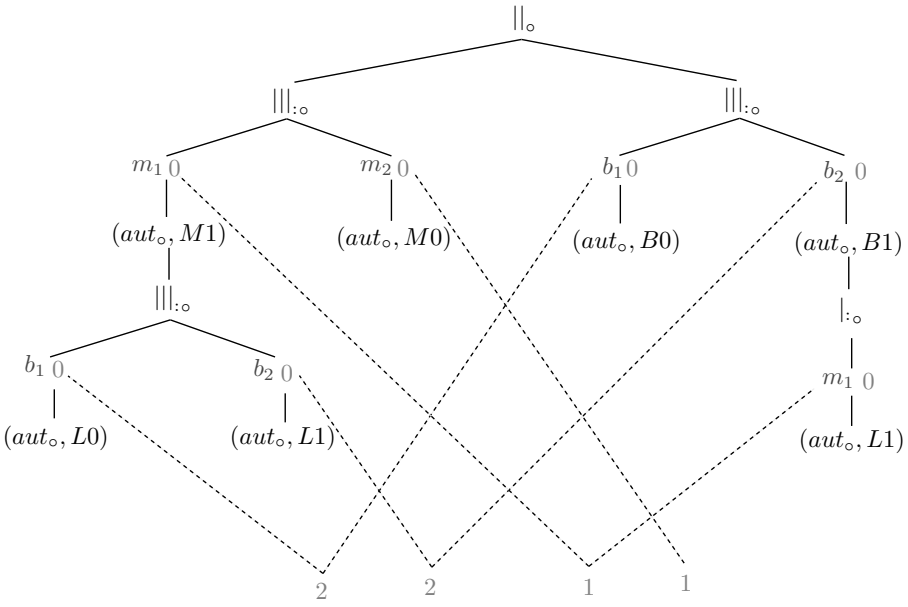

Fig. 10. The graph representation of the state from Figure 3

satisfying the wqo property. But first, we propose another representation of the state. Indeed, we aim at using a wqo theorem from [Din92] (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 labels if we keep the same tree structure (that is also why we cannot use Kruskal's wqo theorem on trees [Kru60]). 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 $(R_1, \ldots, R_n) = val(s)$, let $g(s) = (V', E', \lambda')$ be defined as follows:

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

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) is 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 10 which shows the graph corresponding to the state of Figure 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.

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

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 $v_1, v_2, \ldots, v_n \in V$ such that $\{v_i, v_{i+1}\} \in E$ for all $i \in 1..n-1$. A *simple path* is a path $v_1, v_2, \ldots, v_n \in V$ 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.

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 [Din92] 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 [Din92]). 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 $A = (F, \{T_1, \ldots, T_n\}, \{P_1, \ldots, P_n\})$. 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 a wqo.

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

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

Proposition 5. Bounded-PASTDs are WSTSs.

We will show in Section 4.2 that the coverability problem is decidable for Bounded-PASTDs.

3.5. Classes of Bounded-PASTD

Deciding if a PASTD is a Bounded-PASTD is an open problem, thus, in the following we propose two easily recognizable 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 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 $A = (F, \{T_1, \ldots, T_n\}, \{P_1, \ldots, P_n\})$ 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.

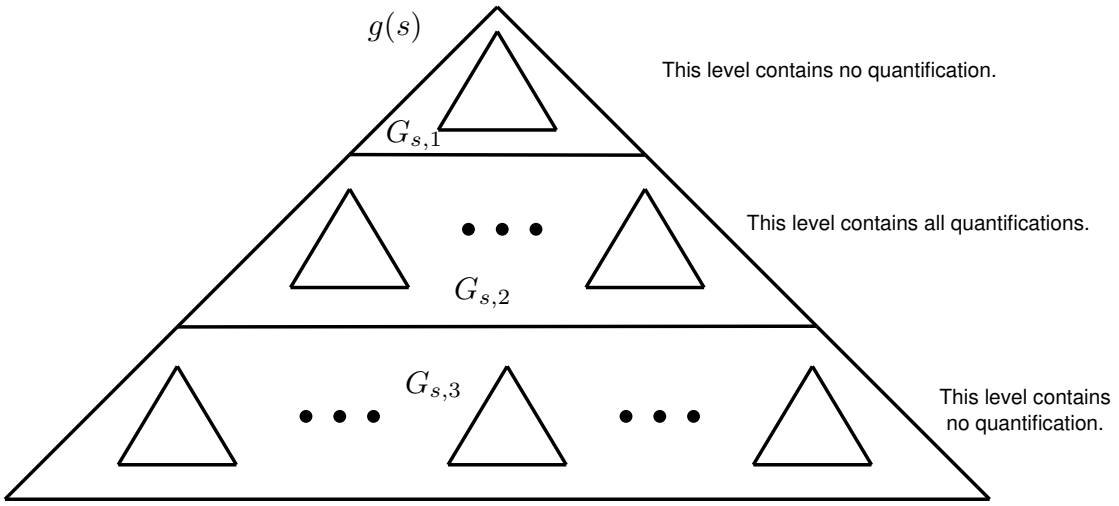

Fig. 11. Node partition of a state

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 $A = (F, \{T_1, \ldots, T_n\}, \{P_1, \ldots, P_n\})$ 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 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 6. Any 1-PASTD is a Bounded-PASTD.

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 $(R_1, \ldots, R_n) = val(s)$ 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 down to 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 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 $\implies$. 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 $\impliedby$. 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 $u = (v_1, \ldots, v_m)$ 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 $u_1, \ldots, u_j$ 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 7. Any Flat-PASTD is a Bounded-PASTD.

Proof. Let $A = (F, \{T_1, \ldots, T_n\}, \{P_1, \ldots, P_n\})$ 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 $W_2 = \{v \in V \mid \lambda(v) \in \{0, 1, \ldots, n\}\}$ 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.

4. Computation of pred-basis for PASTDs

To apply the backward analysis algorithm presented in Section 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 4.1 a new framework which identifies a set of conditions proving the effective pred-basis without wqo. Then, we show in Section 4.2 how to use the framework on PASTDs, hence to solve the coverability problem.

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 of $\uparrow Pred(\uparrow s)$. 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

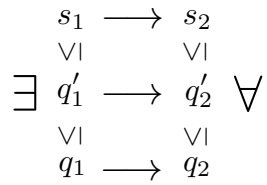

Fig. 12. Condition 4 of RMTS : the backward-downward monotony

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 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) $\forall q'_2 \in Q \cdot q_2 \leq q'_2 \leq s_2 \implies \exists q'_1 \in Q \cdot q'_1 \rightarrow q'_2 \wedge q_1 \leq q'_1 \leq s_1$.

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 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 $s_1 \rightarrow s_2$ is "destructive", that is some instances of entities are lost during the transition and $\gamma(s_2) < \gamma(s_1)$. If it is a "reset transition" (*i.e.* the number of lost instances for similar transitions is unbounded), then there must exist a smaller configuration $s'_1 \leq s_1$ whose rank is bounded by $\gamma(s_2) + d$ and that can fire the transition. If the transition relation never decreases the rank by more than d, then the property is always satisfied.
4. The central property is Condition 4. It 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 exist intermediate transitions. See Figure 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 intermediate transition is also possible.

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 $(q, r) \in Q \times \mathbb{N}^n$.

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 $F : (P \times T \cup T \times P) \rightarrow \mathbb{N}$ a multiset of arcs. A marking is a multiset of places, *i.e.* a mapping $M : P \rightarrow \mathbb{N}$. We denote by $\mathcal{S}_N = (Q, \rightarrow, \subseteq)$ 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 tokens 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 except that the number of tokens 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 $c_1, \ldots, c_n$ 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 $(q, w_1, \ldots, w_n) \leq (q', w_1', \ldots, w_n')$ 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 $\gamma(q, w_1, \ldots, w_n) = (|w_1|, \ldots, |w_n|)$.

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 $\gamma(s) \leq \gamma(s') + (1, \ldots, 1)$.
4. For each type of transition we have a minimal one satisfying the bdm condition. If it is a send action $(q, w_1, \ldots, w_i, \ldots, w_n) \to (q', w_1, \ldots, w_i.a, \ldots, w_n)$, then take the smaller transition $(q, \epsilon, \ldots, \epsilon) \to (q', \epsilon, \ldots, a, \ldots, \epsilon)$. If it is a receive action $(q, w_1, \ldots, a.w_i, \ldots, w_n) \to (q', w_1, \ldots, w_i, \ldots, w_n)$, then take $(q, \epsilon, \ldots, a, \ldots, \epsilon) \to (q', \epsilon, \ldots, \epsilon, \ldots, \epsilon)$. Finally, if it is a loss of message $(q, w_1, \ldots, a.w_i, \ldots, w_n) \to (q, w_1, \ldots, w_i, \ldots, w_n)$, then take $(q, \epsilon, \ldots, a, \ldots, \epsilon) \to (q, \epsilon, \ldots, \epsilon, \ldots, \epsilon)$. Condition 4 is then verified for $c = (1, \ldots, 1)$.

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

Nicely Sliceable WSTS (NSW)[BH05] share some similarities with RMTSs. They both use the same notion of partitioning of the state space and downward monotony. However, they rely on different assumptions and some of their requirements are unnecessary for our purpose. In [BH05], 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 [BH05].

Definition 16 ([BH05]). 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 $q_0 \leq \cdots \leq q_l = q$ 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 exist 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 [BH05] 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 rank 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 allows 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 of $\uparrow Pred(\uparrow s)$, the idea is to compute a finite subset $S \subset \uparrow s$, then 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 $\uparrow_c s$ restrict the set of predecessors and the upward-closure of s to the states that are 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 $p \leq p' \wedge \exists s'' \in Q \cdot s \leq s'' \wedge p \to s'' \wedge \gamma(s'') \leq \gamma(s) + c \wedge \gamma(p) \leq \gamma(s'') + d$, which entails Definition 17 and $p' \in \uparrow Pred_d(\uparrow_c s)$. By the bdm, there exists $q_1 \to q_2$ such that $q_2 \leq s'$, $\gamma(q_2) \leq c$, and $\forall q_2' \in Q \cdot q_2 \leq q_2' \leq s' \implies \exists q_1' \in Q \cdot q_1' \to q_2' \wedge q_1 \leq q_1' \leq p'$. By Condition 2, there exists s'' an upper bound of s and q_2 such that $s'' \leq s'$. Thus have $q_2 \leq s'' \leq s'$. By the bdm, there is $p'' \in Q$ such that $q_1 \leq p'' \leq p'$ and $p'' \to s''$. By Condition 3, there exists $p \in Q$ 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 13. □

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

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

Proof. $Pred_d(\uparrow_c s)$ is a basis of $\uparrow Pred(\uparrow s)$ 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 8. Effective RMTSs have effective pred-basis.

Proof. Let $\mathcal{S} = (Q, \to, \leq, \gamma, c, d)$ 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 $\uparrow_c s = \{q \in \uparrow s \mid \gamma(q) \leq$

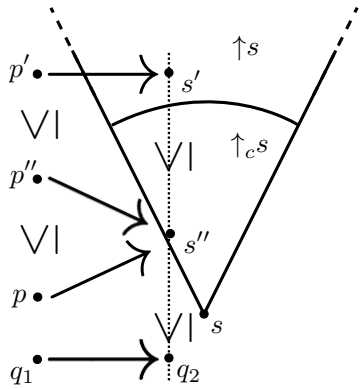

Fig. 13. Illustration of Lemma 6

$\gamma(s)+c\}$, 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 $Pred_d(q) = \{q' \in Pred(q) \mid \gamma(q') \leq \gamma(q) + d\} = \{q' \in \bigcup_{r\leq\gamma(q)+d} \gamma^{-1}(r) \mid q' \to q\}$. □

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 $\mathcal{S}_N = (Q, \to, \subseteq, \gamma, c, d)$ 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 systems [KS14, Mey09] 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 in Section 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 9. For an effective RMTS $\mathcal{S} = (Q, \to, \leq, \gamma, c, d)$ 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 8. Since $\leq$ is a wqo, then $\mathcal{S}$ is a WSTS. Thus, by Theorem 1, the coverability problem is decidable. □

Remark that if we do not have a wqo, we can still use the procedure computing $Pred^*(\uparrow s)$ of Section 3.2 as a semi-algorithm. Furthermore, like in [BH05], we have intermediate results by restricting ourselves to the verification for some specific ranks. Let $\mathcal{S} = (Q, \to, \leq, \gamma, c, d)$ be an effective RMTS. Then, we can always consider a cut-off value $r \in \mathbb{N}^n$ and a subsystem $\mathcal{S}_r = (Q_r, \to, \leq)$ whose set of states is reduced to $Q_r = \{q \in Q \mid \gamma(q) \leq r\}$. During the computing of a basis of $Pred^*(\uparrow s)$, 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 3.2 either by prioritizing lower ranks or by keeping track of computed paths.

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 they 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 [ES96]. In our case the symmetries entailed by the equivalence relation allow us to consider the coverability problem on the quotient space. In order to satisfy condition 2 of Definition 14 we also need to enlarge PASTD states with the size of parameters used to instantiate the PASTD to which a state belongs. Accordingly, the ordering relation is adjusted to take into account these parameters. For bounded PASTD, this extended order is also a wqo. We first state the monotonicity of $(\mathcal{T}_A, \rightarrow, \sim)$, which is necessary for the well-definedness of the quotient state space transitions.

Proposition 10. Let A be a PASTD. Then, $(\mathcal{T}_A, \rightarrow, \sim)$ is monotone.

Proof. The proof is done by induction the structure of states, and similar to the one for $(\mathcal{T}_A, \rightarrow, \preceq)$ (Proposition 3). It stems from the fact that permutation preserves equality and that an ASTD cannot test a particular value. □

Let the point-wise extension of finite set cardinality to vectors of finite sets be noted as $|(R_1, \ldots, R_n)| = (|R_1|, \ldots, |R_n|)$

Definition 18 (Quotient Instantiated ASTD state). Let A be a PASTD with n parameters. Let $Q \subseteq \mathcal{T}_A$. We denote by $Q^\sharp = \{(\tilde{s}, u) \mid s \in Q \wedge u \in \mathbb{N}^n \wedge |val(s)| \leq u\}$. Let $(\tilde{s_1}, u_1)$, $(\tilde{s_2}, u_2) \in \mathcal{T}_A^\sharp$. We say that $(\tilde{s_1}, u_1) \preceq^\sharp (\tilde{s_2}, u_2)$ iff $s_1 \preceq s_2$ and $u_1 \leq u_2$. Let (x_1, u_1), $(x_2, u_1) \in \mathcal{T}_A^\sharp$. We say that $(x_1, u_1) \rightarrow^\sharp (x_2, u_1)$ iff $\exists s_1 \in x_1, s_2 \in x_2 \cdot s_1 \rightarrow s_2$.

We denote by $\mathcal{S}_A^\sharp = (\mathcal{T}_A^\sharp, \rightarrow^\sharp, \preceq^\sharp)$ the quotient system.

Proposition 11. Let A be a PASTD. Then, $\mathcal{S}_A^\sharp$ is monotone.

Proof. The proof stems easily from Definition 18 and Proposition 3. □

We need the following lemma to prove that the coverability problem in $\mathcal{S}_A$ can be reduced to the coverability problem in $\mathcal{S}_A^\sharp$.

Lemma 8. If $(x_1, u_1) \rightarrow^\sharp (x_2, u_1)$, then $\forall s_1 \in x_1 \cdot \exists s_2 \in x_2 \cdot s_1 \rightarrow s_2$.

Proof. The proof stems easily from Definition 18 and Proposition 10. □

Theorem 4. Let A be a PASTD. Then, $cov(\mathcal{S}_A, I, s) \iff cov(\mathcal{S}_A^\sharp, I^\sharp, (\tilde{s}, |val(s)|))$.

Proof. Direction $\Rightarrow$: Let $i \xrightarrow{*} s'$ with $i \in I$ and $s \preceq s'$ be a run of $\mathcal{S}_A$. Let

$$u = sup(\{|val(q)| \mid q \text{ is a state of } i \xrightarrow{*} s'\}).$$

Let $q \rightarrow q'$ be a step of than run. By definition of $\rightarrow^\sharp$, we have $(\tilde{q}, u) \rightarrow^\sharp (\tilde{q'}, u)$. Thus, we have $(\tilde{i}, u) \xrightarrow{*}_\sharp (\tilde{s'}, u)$ and $(\tilde{s}, |val(s)|) \preceq^\sharp (\tilde{s'}, u)$. For other direction ($\Leftarrow$): let $(x_1, u') \xrightarrow{*}_\sharp (x_m, u')$ with $x_1 \in I^\sharp$ and $(\tilde{s}, |val(s)|) \preceq^\sharp (x_m, u')$ be a run of $\mathcal{S}_A^\sharp$. We can construct a run $s_1 \xrightarrow{*} s_m$ such that $s \preceq s_m$. By definition of $\rightarrow^\sharp$, there exists $s_1', s_2' \cdot s_1' \in x_1 \wedge s_2' \in x_2 \wedge s_1' \rightarrow s_2'$. Take $s_1 = s'1$ and $s_2 = s_2'$. Since $x_1 \in I^\sharp$ and $s_1 \in x_1$, then $s_1 \in I$. By Lemma 8, we can construct each s_j for $j \in 3..m-1$ such that $s_j \rightarrow s_{j+1}$. Since $s_m \in x_m$ and $(\tilde{s}, |val(s)|) \preceq^\sharp (x_m, u')$, then by definition of $\preceq^\sharp$ we have $s \preceq s_m$. □

Similarly, the reachability problem in $\mathcal{S}_A$ si equivalent to the reachability problem in $\mathcal{S}_A^\sharp$ and we can state the following result.

Proposition 12. Reachability is undecidable for PASTDs.

Proof. From Proposition 1, PASTDs simulates RVASSs. Hence, for every RVASS, there exist a PASTD A and an injection from the RVASS to $\mathcal{S}_A^\sharp$ preserving the transitions. Reachability of configurations is undecidable for RVASSs, thus reachability of equivalence classes is undecidable for PASTDs. □

From now on, for a PASTD A we study the coverability problem on the quotient MTS $\mathcal{S}_A^\sharp$. The objective is to find an RMTS for A. Let us define an adequate rank function.

Definition 19. For a PASTD A, let $\gamma : \mathcal{T}_A^\sharp \rightarrow \mathbb{N}^n$ be defined by $\gamma((\tilde{s}, u)) = u$.

A state s of $\mathcal{T}_A$ can belong to any instantiation of a PASTD such that $val(s) \subseteq \overrightarrow{U}$. The function γ gives for a particular instantiation of state s the size of the instantiation parameters. All states of an equivalence class in $\mathcal{T}_A^\sharp$ have the same size. For instance, the state of Figure 3 from the library example can be from a PASTD instantiated with $\overrightarrow{U} \supseteq (\{m_1, m_2\}, \{b_1, b_2\})$. The state describes a configuration with at least two members and two books.
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 9. Let A be a PASTD. For all $x_1, x_2, x_3 \in \mathcal{T}_A^\sharp$ such that $x_1 \preceq^\sharp x_3$ and $x_2 \preceq^\sharp x_3$, there exists $x_4 \in \mathcal{T}_A^\sharp$ such that $x_1 \preceq^\sharp x_4$, $x_2 \preceq^\sharp x_4$, $x_4 \preceq^\sharp x_3$ and $\gamma(x_4) \leq \gamma(x_1) + \gamma(x_2)$.

Proof. Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$. Let $x_1 = (\tilde{s_1}, u_1)$, $x_2 = (\tilde{s_2}, u_2)$, $x_3 = (\tilde{s_3}, u_3)$. 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 Definition 18, $s_1 \preceq s_3$; by Definition 9, 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 $x_4 = (\tilde{s_4}, u_4)$ as follows Let $s_4 = (V_4, E_4, \lambda_4)$ such that $V_4 = \{v \in V_3 \mid v \in Im(f_{13}) \cup Im(f_{23})\}$, $E_4 = \{\{v, w\} \in E_3 \mid v, w \in V_4\}$ 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 $s_2 \preceq s_4$. Let $Z = \{|val(s_4)|, u_1, u_2\}$ and $u_4 = sup(Z)$. Since u_3 is an upper bound of Z, we have $u_4 \leq u_3$, hence $(\tilde{s_1}, u_1), (\tilde{s_2}, u_2) \preceq^\sharp (\tilde{s_4}, u_4) \preceq^\sharp (\tilde{s_3}, u_3)$. We have that $|val(s_4)| \leq |val(s_1)| + |val(s_2)|$ 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 $\sigma_{23,i}(\lambda_2(w)) = \lambda_4(v)$. For each $i \in 1..n$: if $u_4[i] = |val(s_4)[i]|$ then $u_4[i] \leq |val(s_1)[i]| + |val(s_2)[i]| \leq u_1[i] + u_2[i]$, by Definition 18; if $u_4[i] = u_1[i]$ or $u_2[i]$, then $u_4[i] \leq u_1[i] + u_2[i]$. Hence $\gamma(x_4) \leq \gamma(x_1) + \gamma(x_2)$. □

Intuitively, take the example of the library and consider a state x_1 such that $\gamma(x_1) = (1, 1)$ and where the member m_1 borrowed the book b_1 and a state x_2 such that $\gamma(x_2) = (1, 1)$ and where the member m_2 has returned the book b_2, then clearly x_1 and x_2 have upper bounds. We can construct a "little" state x_3 such that $\gamma(x_3) = (2, 2)$ 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 the function μ defined as follows.

Definition 20. 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. $\mu(\epsilon) = (0, \ldots, 0)$
2. $\mu(\mathcal{A}[q_1[F_1], \ldots, q_j[F_j]]) = sup(\{\mu(F_i) \mid i \in 1..j\})$
3. $\mu(||_\Delta[F_1, F_2]) = \mu(F_1) + \mu(F_2)$
4. $\mu(|_{x \in T_i}[F]) = \mu(F) + (u_1, \ldots, u_n)$, where $u_i = 1$ and $\forall i \neq j \cdot u_j = 0$
5. $\mu(|||_{x \in T_i}[F]) = \mu(F) + (u_1, \ldots, u_n)$, where $u_i = 1$ and $\forall i \neq j \cdot u_j = 0$

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 is 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 10. Let $A = (F, \overrightarrow{T}, \overrightarrow{P})$ be a PASTD and $c = \mu(F)$. For each transition $x_1 \rightarrow^\sharp x_2$, there is $y_1 \rightarrow^\sharp y_2$ such that $y_2 \preceq^\sharp x_2$, $\gamma(y_2) \leq c$, and $\forall z_2 \in \mathcal{T}_A^\sharp \cdot y_2 \preceq z_2 \preceq x_2 \implies \exists z_1 \in \mathcal{T}_A^\sharp \cdot z_1 \rightarrow z_2 \wedge y_1 \preceq z_1 \preceq x_1$.

Proof. By induction on the structure F of the PASTD. See a similar proof in [CYF17]. □

For instance, consider the PASTD of the library $(F, (T_m, T_b), (P_m, P_b))$ whose expression F is illustrated in Figure 2. By Definition 20, 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 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 $\mathcal{S}^\sharp_A = (\mathcal{T}^\sharp_A, \to^\sharp, \preceq^\sharp)$ the quotient MTS, then $\mathcal{S} = (\mathcal{T}^\sharp_A, \to^\sharp, \preceq^\sharp, \gamma, \mu(F), \overrightarrow{0})$ is an effective RMTS.

Proof. $(\mathcal{T}^\sharp_A, \to^\sharp, \preceq^\sharp)$ is a MTS by Proposition 11 and $\preceq$ a po on $\mathcal{T}^\sharp_A$.

1. Let $x, y \in \mathcal{T}^\sharp_A$ such that $x \preceq^\sharp y$. 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 $x = (\tilde{s}, r) \in \gamma^{-1}(r)$ with $\tilde{g}(s) = (V, E, \lambda)$, $|V| \leq k$. Then, for all $x = (\tilde{s}, r) \in \gamma^{-1}(r)$ with $\tilde{g}(s) = (V, E, \lambda)$, $|V|$ 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 9.
3. For all $x \to^\sharp y$, we have $\gamma(x) = \gamma(y)$ by Definitions 18 and 19. Thus $\gamma(x) \leq \gamma(y) + \overrightarrow{0}$.
4. Condition 4 is proved by Lemma 10.

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

For any PASTD, we conclude that we can compute a finite pred-basis of any state x. An alternative to enumerating $\gamma^{-1}(r)$ for computing the finite pred-basis $Pred_d(\uparrow_c x)$ is to use the successors wrt $\preceq^\sharp$. For a PASTD, $succ(x) = \{x' \mid x \preceq^\sharp x' \wedge \neg\exists x'' \cdot x \preceq^\sharp x'' \preceq^\sharp x'\}$ is computable. Thus, we can iterate over the successors x' until $\gamma(x') > \gamma(x) + c$ and compute their predecessors s'' wrt $\to^\sharp$ until $\gamma(x'') > \gamma(x') + d$. $Pred(x')$ can be infinite for a PASTD, but it can be enumerated by starting with the smallest predecessors and incrementing their ranks until $\gamma(x'') > \gamma(x') + d$. If we want to compute a finite basis of $Pred^*(\uparrow x)$ and decide coverability, we need the wqo condition.

Proposition 13. Let A be a Bounded-PASTD. $(\mathcal{T}^\sharp_A, \preceq^\sharp)$ is a wqo.

Proof. By Theorem 3, $(\mathcal{T}_A, \preceq)$ is a wqo; thus the quotient $\mathcal{T}_A/\sim$ is also a wqo. $(\mathbb{N}^n, \leq)$ is also a wqo (Dickson's lemma). Thus, $(\mathcal{T}^\sharp_A, \preceq^\sharp)$ is a Cartesian product of wqos. The Cartesian product of wqos is a wqo. □

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, $(\mathcal{T}^\sharp_A, \to^\sharp, \preceq^\sharp, \mu(F), \overrightarrow{0})$ the corresponding RMTS, $I^\sharp \subseteq \mathcal{T}^\sharp_A$ and $x \in \mathcal{T}^\sharp_A$. We have that $\preceq^\sharp$ is a wpo as A is a Bounded-PASTD, thus we can compute a basis of $Pred^*(\uparrow x)$. We can check whether $Pred^*(\uparrow x) \cap I^\sharp$ is empty or not because $I^\sharp$ is easily recognisable (initial states are explicited in Appendix A). Hence, by Proposition 9, $cov(\mathcal{S}^\sharp_A, I^\sharp, x)$ is decidable. □

5. Related Work

Several models in the literature allow for the specification of parameterized systems [Pet81, HSBR10, SK09b, DSZ10, KS14], 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 [BK84, BB87, FSD03] are suitable to specify information systems but it is not always clear how to handle parameterized systems. In particular, [FFC+10] 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 [VLDM16] but the paper does not consider parameters. In [SK09b], 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 [McM99, CTV06, AHH13] and cut-off techniques [EK00, HSBR10, SK09b, SK09a, KKW10, AHH13], well-structured transition systems were widely used to verify parameterized systems [Mey09, KS14, DSZ10, AHH13] or more generally infinite systems. Finding the right wqo on the set of states is a central issue. In [DSZ10, KS14], 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. The work of [Mey09] studies π-Calculus expressions, a class a process algebras. For a subclass of π-Calculus expressions, called structurally-stationary dynamically reconfigurable systems, the author proposes a qo based on a rooted tree ordering on process fragments, and shows that processes of bounded depth form a WSTS. Fragments are based on sequential processes which are related by the restriction operator. Depth boundedness is based on the nesting of restriction expressions within a fragment, whereas in our case boundedness arises from simple paths whose length depends on occurrences of quantified variables values in a state, which are more difficult to characterise syntactically. We provide two syntactical subclasses of Bounded-PASTDs. Alternate wqos on graphs, like the *graph minor* relation [RS10], can also be used [KS14]. But in our case it does not satisfy the monotony condition.
In [KS14], 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* [BH05], which is closer to ours in term of requirements, gives also a subclass of WSTS. However, they are both tied to a wqo condition (by definition of WSTS), unlike the RMTS framework.

6. Conclusion

Results and discussion. In this article, we 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 which 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) which 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 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 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 [CYF17] 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 of $Pred^*(\uparrow s)$ 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 [SS12]. 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

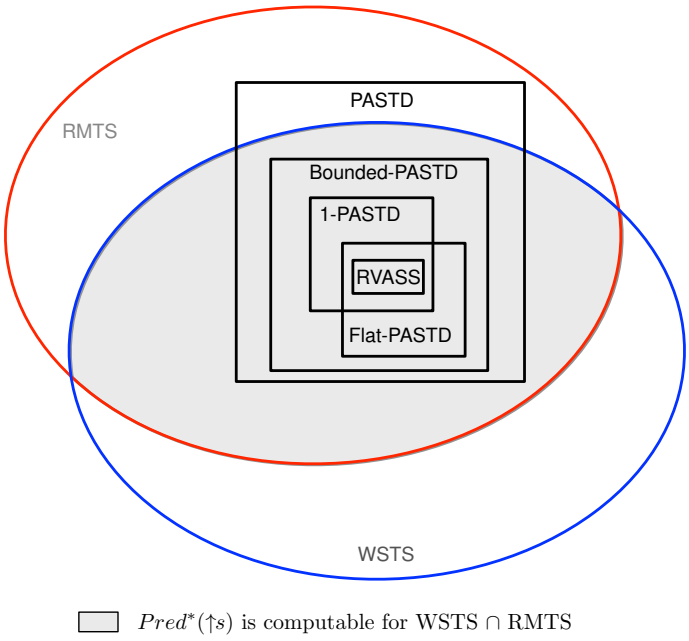

Fig. 14. Comparison of the models

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.

References

[ACJT96] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Logic in Computer Science*, pages 313–321. IEEE, 1996.

[AHH13] Parosh Aziz Abdulla, Frédéric Haziza, and Lukáš Holík. All for the price of few. In *Verification, Model Checking, and Abstract Interpretation*, volume 7737 of *LNCS*, pages 476–495. Springer, 2013.

[BB87] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1987.

[BH05] 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*, volume 3440 of *LNCS*, pages 77–92. Springer, 2005.

[BK84] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Information and Control*, 60(1):109–137, 1984.

[BSM99] P. Bernus, G. Schmidt, and K. Mertins, editors. *Handbook on Architectures of Information Systems*. Springer, 1999.

[CTV06] Edmund Clarke, Muralidhar Talupur, and Helmut Veith. Environment abstraction for parameterized verification. In *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *LNCS*, pages 126–141. Springer, 2006.

[CYF17] Raphaël Chane-Yack-Fa. Verification of parameterized algebraic state transition diagrams. Technical report, Département d'informatique, Faculté des Sciences, Université de Sherbrooke, 2017. `http://info.usherbrooke.ca/mfrappier/Papers/pastd.pdf`.

[DFS98] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In *Automata, Languages and Programming*, volume 1443 of *LNCS*, pages 103–115. Springer, 1998.

[Din92] Guoli Ding. Subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 16(5):489–502, 1992.

[DSZ10] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *Concurrency Theory*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.

[EJFG+10] 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 *International Conference on Enterprise Information Systems*, pages 374–379. SciTePress, 2010.

[EK00] E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In *Automated Deduction*, volume 1831 of *LNCS*, pages 236–254. Springer, 2000.

[ES96] E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1-2):105–131, 1996.

[FFC+10] 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*, volume 6447 of *LNCS*, pages 581–596. Springer, 2010.

[FGL+08] Marc Frappier, Frédéric Gervais, Régine Laleau, Benoît Fraikin, and Richard St-Denis. Extending statecharts with process algebra operators. *Innovations in Systems and Software Engineering*, 4(3):285–292, 2008.

[FGLF08] Marc Frappier, Frédéric Gervais, Régine Laleau, and Benoît Fraikin. Algebraic state transition diagrams. Technical report, Université de Sherbrooke, 2008. `http://info.usherbrooke.ca/mfrappier/Papers/astd.pdf`.

[Fin87] Alain Finkel. A generalization of the procedure of karp and miller to well structured transition systems. In *Automata, Languages and Programming*, volume 267 of *LNCS*, pages 499–508. Springer, 1987.

[Fin94] Alain Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3):129–135, 1994.

[FS01] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63–92, 2001.

[FSD03] Marc Frappier and Richard St-Denis. EB^3: an entity-based black-box specification method for information systems. *Software and Systems Modeling*, 2(2):134–149, 2003.

[Har87] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[Hig52] Graham Higman. Ordering by divisibility in abstract algebras. In *Proceedings of the London Mathematical Society*, volume s3-2, pages 326–336, 1952.

[Hoa78] Charles A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[HP79] John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979.

[HSBR10] Youssef Hanna, David Samuelson, Samik Basu, and Hridesh Rajan. Automating cut-off for multi-parameterized systems. In *Formal Methods and Software Engineering*, volume 6447 of *LNCS*, pages 338–354. Springer, 2010.

[KKW10] Alexander Kaiser, Daniel Kroening, and Thomas Wahl. Dynamic cutoff detection in parameterized concurrent programs. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 645–659. Springer, 2010.

[Kru60] Joseph B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, pages 210–225, 1960.

[KS14] Barbara König and Jan Stückrath. A general framework for well-structured graph transformation systems. In *Concurrency Theory*, volume 8704 of *LNCS*, pages 467–481. Springer, 2014.

[McM99] Kenneth L. McMillan. Verification of infinite state systems by compositional mmodel checking. In *Correct Hardware Design and Verification Methods*, volume 1703 of *LNCS*, pages 219–234. Springer, 1999.

[Mey09] Roland Meyer. *Structural Stationarity in the π-Calculus*. PhD thesis, Department für Informatik, Carl von Ossietzky Universität, Oldenburg, 2009.

[Mil89] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Pet81] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1981.

[RHB97] A. W. Roscoe, C. A. R. Hoare, and Richard Bird. *The Theory and Practice of Concurrency*. Prentice Hall, 1997.

[RS10] Neil Robertson and Paul D. Seymour. Graph minors xxiii. nash-williams' immersion conjecture. *Journal of Combinatorial Theory*, 100(2):181–205, 2010.

[SK09a] Antti Siirtola and Juha Kortelainen. Algorithmic verification with multiple and nested parameters. In *Formal Methods and Software Engineering*, volume 5885 of *LNCS*, pages 561–580. Springer, 2009.

[SK09b] Antti Siirtola and Juha Kortelainen. Parameterised process algebraic verification by precongruence reduction. In *Application of Concurrency to System Design*, pages 158–167. IEEE, 2009.

[SS12] Sylvain Schmitz and Philippe Schnoebelen. Algorithmic aspects of wqo theory. Lecture Notes, 2012.

[VLDM16] Dimitris Vekris, Frédéric Lang, Catalin Dima, and Radu Mateescu. Verification of eb^3 specifications using CADP. *Formal Asp. Comput.*, 28(1):145–178, 2016.

A. Operational semantics of ASTDs

Some states of an ASTD may be initial. This is specified by a function *init* that returns the initial state of an ASTD.

Definition 21 (Initial states). Let F be an ASTD expression. The function *init*, which returns the initial state of F, is defined as follows:

1. $init(\mathcal{A}) = (\mathsf{aut}_\circ, q_i)$, where q_i is the initial state of $\mathcal{A}$
2. $init(\mathcal{A}[q_1[F_1], \ldots, q_k[F_k]]) = (\mathsf{aut}_\circ, q_i)[init(F_i)]$, where q_i is the initial state of $\mathcal{A}$
3. $init(||_\Delta[F_1, F_2]) = ||_\Delta[init(F_1), init(F_2)]$
4. $init(\,|_{x \in T}[F]) = |:_\circ$
5. $init(\,|||_{x \in T}[F]) = |||:_\circ[p_1[init(F)], \ldots, p_k[init(F)]]$, where $T = \{p_1, \ldots, p_k\}$

As we also consider abstract states, we introduce a function *Abinit* which characterizes the set of abstract initial states.

Definition 22 (Abstract initial states). Let F be an ASTD expression. We define the set $Abinit(F)$ by:

$$Abinit(F) = \{s \mid s \preceq init(F)\}$$

Similarly to initial states, we define final states by a predicate *final* that determines whether a state is considered as final.

Definition 23 (Final states). Let F be an ASTD expression and s a state of F. The predicate $final_F(s)$, which determines if s is final in F, is defined as follows:

1. If $s = (\mathsf{aut}_\circ, q)$ and $F = \mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$, then $final_F(s) \equiv q \in Q_f$
2. If $s = (\mathsf{aut}_\circ, q)[s']$ and $F = \mathcal{A}[q_1[F_1], \ldots, q[F'], \ldots, q_k[F_k]]$ with $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$, then $final_F(s) \equiv q \in Q_f \wedge final_{F'}(s')$
3. If $s = ||_\circ[s_1, s_2]$ and $F = ||_\Delta[F_1, F_2]$, then $final_F(s) \equiv final_{F_1}(s_1) \wedge final_{F_2}(s_2)$
4. If $s = |:_\circ$ and $F = |_{x \in T}[F']$, then $final_F(s) \equiv final_{F'}(init(F'))$
5. If $s = |:_\circ[p[s']]$ and $F = |_{x \in T}[F']$, then $final_F(s) \equiv final_{F'}(s')$
6. If $s = |||:_\circ[p_1[s_1], \ldots, p_k[s_k]]$ and $F = |||_{x \in T}[F']$, then $final_F(s) \equiv \exists i \leq k \cdot final_{F'}(s_i)$

Automaton ASTDs 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 (Environment). 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, $u[\Gamma_1 \lhd \Gamma_2] = (u[\Gamma_1])[\Gamma_2]$ (here, brackets denote a substitution). 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 $book(b)$ from Figure 1 where b is instantiated by a value b_1, and we look locally at the sub-state $(\mathsf{aut}_\circ, B1)[|:_\circ[m_1[(\mathsf{aut}_\circ, L0)]]]$, we need the environment $(\!|b := b_1|\!)$ in the sub-ASTD $loan(b_1)$ to compute the transition $lend(b_1, m_1)$, 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 predicate *final*. However, it is important in the definition of the transition relation of ASTDs. The operational semantics of ASTDs consists of a set of inference rules defined inductively. We write transitions with respect to an environment Γ, noted as $s \xrightarrow{\sigma,\Gamma}_F s'$, where s and s' are two states of an ASTD F and σ an event. Subscript F can be omitted when it is clear from the context which ASTD is being referred to.

Definition 25 (Transition rules). Let F be an ASTD expression and s and s' two states of F. 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 F be an ASTD, Γ an environment and σ an event. The operational semantics is inductively defined for each ASTD subtype.

1. If $F = \mathcal{A}[q_1[F_1], \ldots, q_k[F_k]]$ with $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$, each transition in F is given by one of the two following inference rules:

$$\mathsf{aut}_1\ \frac{\delta(q_i, q_j, \sigma', final?) \qquad final? \Rightarrow final_{F_i}(s) \qquad \sigma'[\Gamma] = \sigma \qquad s' \in Abinit(F_j)}{(\mathsf{aut}_\circ, q_i)[s] \xrightarrow{\sigma,\Gamma}_F (\mathsf{aut}_\circ, q_j)[s']}$$

$$\mathsf{aut}_2\ \frac{s \xrightarrow{\sigma,\Gamma}_{F_i} s'}{(\mathsf{aut}_\circ, q_i)[s] \xrightarrow{\sigma,\Gamma}_F (\mathsf{aut}_\circ, q_i)[s']}$$

Rule aut_1 describes a transition between local states. There is a transition, labeled by σ and with environment Γ, between some state of q_i and the initial state of q_j if there is an arrow in the automaton between q_i and q_j, 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 $F = ||_\Delta[F_l, F_r]$,

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

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

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.

3. If $F = |_{x \in T}[F']$,

$$|:_1\ \frac{s \xrightarrow{\sigma,(\![x:=p]\!)\triangleleft\Gamma}_{F'} s' \qquad s \in Abinit(F') \qquad p \in T}{|:_\circ \xrightarrow{\sigma,\Gamma}_F |:_\circ[p[s']]}$$

$$|:_2\ \frac{s \xrightarrow{\sigma,(\![x:=p]\!)\triangleleft\Gamma}_{F'} s'}{|:_\circ[p[s]] \xrightarrow{\sigma,\Gamma}_F |:_\circ[p[s']]}$$

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.

4. If $F = |||_{x \in T}[F']$,

$$|||:_1\ \frac{s \xrightarrow{\sigma,(\![x:=p_i]\!)\triangleleft\Gamma}_{F'} s'}{|||:_\circ[p_1[s_1], \ldots, p_i[s], \ldots, p_k[s_k]] \xrightarrow{\sigma,\Gamma}_F |||:_\circ[p_1[s_1], \ldots, p_i[s'], \ldots, p_k[s_k]]}$$

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