Forward Analysis of Depth-Bounded Processes

Thomas Wies, Damien Zufferey, and Thomas A. Henzinger

IST Austria (Institute of Science and Technology Austria)

Abstract. Depth-bounded processes form the most expressive known fragment of the π-calculus for which interesting verification problems are still decidable. In this paper we develop an adequate domain of limits for the well-structured transition systems that are induced by depth-bounded processes. An immediate consequence of our result is that there exists a forward algorithm that decides the covering problem for this class. Unlike backward algorithms, the forward algorithm terminates even if the depth of the process is not known a priori. More importantly, our result suggests a whole spectrum of forward algorithms that enable the effective verification of a large class of mobile systems.

1 Introduction

We are interested in the verification of π-calculus processes [21,22], i.e., message passing systems that admit unbounded creation of processes and name mobility. We can think of a configuration of such a system as a graph [14, 20]. The vertices of the graph are the processes labelled by their current local state. Edges between processes indicate whether the respective processes share a channel, i.e., whether they are able to communicate with each other.

The most expressive known fragment of the π-calculus for which interesting verification problems are still decidable is the class of depth-bounded processes [18]. Intuitively, in a depth-bounded process there is a bound on the length of all simple paths in all reachable configuration graphs (the graphs may contain cycles). A typical example of a depth-bounded process is a server-client architecture where a server answers requests of clients and where each client only knows the name of the server but not the names of other clients. Both the number of simultaneously active clients as well as the number of pending requests for the server can be unbounded.

In this paper we are concerned with the covering problem for depth-bounded processes. Intuitively, the covering problem asks whether a system can reach a configuration that contains some process that is in a local error state. A decision procedure for the covering problem therefore enables the automated verification of an interesting class of safety properties. Meyer showed in [18] that depth-bounded processes admit well-structured transition systems (WSTS) [1, 9, 12]. This implies that the covering problem for depth-bounded processes of known depth can be decided using a standard backward algorithm for WSTSs. The question whether the covering problem is decidable for the entire class of depth-bounded processes was open.

We present the first forward algorithm for this problem. Unlike backward algorithms, our algorithm terminates even if the bound of the system is not known a priori. We thus show that the covering problem is decidable for the entire class. Our algorithm

is an instance of the expand, enlarge, and check algorithm schema for WSTSs that exhibit a so-called *adequate domain of limits* (ADL) [10, 13]. An adequate domain of limits for the well-quasi-ordering of a WSTS provides an effective representation of all downward-closed sets of configurations, i.e., ADLs are the key for ensuring termination of forward analyses of WSTSs. Our main technical contribution is the development of an adequate domain of limits for depth-bounded processes. For this purpose we show that downward-closed sets of configurations in depth-bounded processes are characterized by finite unions of regular languages of unranked trees.

Besides our theoretical interest in forward analysis of π-calculus processes there are also practical considerations that make forward algorithms more appealing than their backward counterparts. A backward analysis needs to consider all possible unifications between names that may enable processes to synchronize. A forward analysis instead knows which names are equal and which are not. In practice, the search space of a forward analysis is therefore often significantly smaller than the search space of a backward analysis. We give an example that demonstrates this phenomenon in Section 3. While the forward algorithm that we consider in this paper is mainly of theoretical interest, our adequate domain of limits suggests a whole spectrum of forward algorithms that enable the effective verification of a large class of mobile systems. This spectrum ranges from acceleration-based algorithms in the style of Karp-Miller [8, 11, 15] to approximation algorithms based on abstract interpretation [6].

Further related work. Depth-bounded processes are semantically defined in terms of reachable configurations. While checking depth-boundedness is in general undecidable, many fragments of the π-calculus that are defined syntactically [2, 7] or in terms of type systems [4, 25, 26] are subsumed by depth-bounded processes. Our result carries over to these fragments. Further related work can be found in the context of graph rewriting systems. Bauer and Wilhelm [3] developed an overapproximating shape analysis for graph rewriting systems whose reachable configurations have a star-like shape. Such systems are bounded in the length of the acyclic paths. Our result naturally generalizes to such systems and promises complete algorithms for their verification.

2 Preliminaries

We first fix the syntax and semantics of our version of the π-calculus and briefly recall depth-bounded processes, well-quasi-orderings, better-quasi-orderings, and well-structured transition systems.

2.1 The π-Calculus and Depth-Bounded Processes

We consider systems of recursive equations in the polyadic π-calculus that have a specific normal form inspired by Amadio and Meyssonnier [2].

Assume a countable infinite set of names with typical elements x, y and a countable infinite set of process identifiers with typical elements A, B. We assume that each name and identifier has an associated *arity* in $\mathbb{N}$. We denote by $\boldsymbol{x}$ a (possibly empty) vector over names and denote by $[\boldsymbol{x}/\boldsymbol{y}]$ a substitution on names.

Process terms P are recursively composed of the unit process 0, parameterized process identifiers $A(\boldsymbol{x})$, and the standard operations of parallel composition $P_1 \mid P_2$, external choice $\pi_1.P_1 + \pi_2.P_2$, and name restriction $(\nu x)P$. Hereby, a prefix π is either an *input prefix* of the form $x(\boldsymbol{y})$ or an *output prefix* of the form $\overline{x}(\boldsymbol{y})$. All parameter vectors occuring in process terms must respect the arities of names and identifiers. We call the terms of the form $A(\boldsymbol{x})$ *threads*. We write Π in order to denote indexed parallel composition and Σ for indexed external choice. We further write $(\nu \boldsymbol{x})$ for $(\nu x_1) \ldots (\nu x_n)$ where $\boldsymbol{x} = x_1, \ldots, x_n$. An occurrence of a name x in a process term P is called *free* if it is not below a (νx) or an input prefix $y(x)$. We denote by $\mathsf{fn}(P)$ the set of all free occurring names in P. We say that P is *closed* if $\mathsf{fn}(P) = \emptyset$. We denote by $P \equiv Q$ the usual structural congruence relation on process terms, i.e., P is syntactically equal to Q up to renaming and reordering of restricted names, scope extrusion, elimination of units, and associativity and commutativity of parallel composition and external choice.

A *configuration* is a closed process term of the following form

$$(\nu \boldsymbol{x})(\prod_{i \in I} A_i(\boldsymbol{x}_i))$$

A *process* $\mathcal{P}$ is a pair $(I, \mathcal{E})$ where I is an *initial* configuration and $\mathcal{E}$ is a finite set of parametric equations $A(\boldsymbol{x}) = P$ such that (1) every process identifier in P and I is defined by exactly one equation in $\mathcal{E}$ and (2) $\mathsf{fn}(P) \subseteq \{\boldsymbol{x}\}$. We assume that all equations in $\mathcal{E}$ have the following normal form:

$$A(\boldsymbol{x}) = \sum_{i \in I} \pi_i.(\nu \boldsymbol{x}_i)(\prod_{j \in J_i} A_j(\boldsymbol{x}_j))$$

Operational semantics. Given a process $\mathcal{P} = (I, \mathcal{E})$, we define a transition relation $\rightarrow_{\mathcal{E}}$ on configurations that captures the usual π-calculus reduction rules as follows. Let P and Q be configurations then we have $P \rightarrow_{\mathcal{E}} Q$ if and only if the following conditions hold:

1. $P \equiv (\nu \boldsymbol{u})(A(\boldsymbol{v}) \mid B(\boldsymbol{w}) \mid P')$,
2. the defining equation of A in $\mathcal{E}$ is of the form $A(\boldsymbol{x}) = x(\boldsymbol{x'}).(\nu \boldsymbol{x''})(M) + M'$,
3. the defining equation of B in $\mathcal{E}$ is of the form $B(\boldsymbol{y}) = \overline{y}(\boldsymbol{y'}).(\nu \boldsymbol{y''})(N) + N'$,
4. $\sigma = [\boldsymbol{v}/\boldsymbol{x}, \boldsymbol{w}/\boldsymbol{x'}, \boldsymbol{z}_A/\boldsymbol{x''}, \boldsymbol{w}/\boldsymbol{y}, \boldsymbol{z}_B/\boldsymbol{y''}]$ where $\boldsymbol{z} = \boldsymbol{z}_A, \boldsymbol{z}_B$ are fresh names,
5. $\sigma(x) = \sigma(y)$,
6. $Q \equiv (\nu \boldsymbol{u}, \boldsymbol{z})(\sigma(M) \mid \sigma(N) \mid P')$.

We denote by $\rightarrow_{\mathcal{E}}^*$ the reflexive transitive closure of the relation $\rightarrow_{\mathcal{E}}$. We say that a configuration P is *reachable* in process $\mathcal{P}$ if and only if $I \rightarrow_{\mathcal{E}}^* P$. Finally, we denote by $\mathit{Reach}(\mathcal{P})$ the set of all reachable configurations of process $\mathcal{P}$.

Depth-Bounded Processes. We now recall the definition of the class of depth-bounded processes [18]. The *nesting of restrictions* nest_ν of a process term is measured recursively as follows $\mathit{nest}_\nu(0) = \mathit{nest}_\nu(A(\boldsymbol{x})) = \mathit{nest}_\nu(P_1 + P_2) = 0$, $\mathit{nest}_\nu((\nu x)P) = 1 + \mathit{nest}_\nu(P)$, and $\mathit{nest}_\nu(P_1 \mid P_2) = \max\{\mathit{nest}_\nu(P_1), \mathit{nest}_\nu(P_2)\}$. The *depth* of a process term P is the minimal nesting of restrictions of process terms in the congruence class of P:

$$\mathit{depth}(P) = \min\{\, \mathit{nest}_\nu(Q) \mid Q \equiv P \,\}.$$

Definition 1 (Depth-Boundedness). *A set of configurations $\mathcal{C}$ is called* depth-bounded *if there is $k_D \in \mathbb{N}$ such that $depth(P) \leq k_D$ for all $P \in \mathcal{C}$. A process $\mathcal{P}$ is called* depth-bounded *if its set of reachable configurations $Reach(\mathcal{P})$ is depth-bounded.*

Example 2. The following equations describe a simple client-server system where a server can spawn clients and answer their requests. The server is given by process identifier $Server(x, y)$. The channel x is used for communication with clients. The channel y is used to trigger creation of new clients.

$$\begin{aligned} Server(x,y) &= (x(z).Answer(z) \mid Server(x,y)) \\ &\quad + (y().(\nu u)(Client(u,x) \mid Answer(u) \mid New(y) \mid Server(x,y))) \\ Client(u,x) &= u().(Client(u,x) \mid Request(x,u)) \\ Answer(u) &= \overline{u}().0 \qquad New(y) = \overline{y}().0 \qquad Request(x,u) = \overline{x}(u).0 \end{aligned}$$

If the initial configuration is given by $(\nu x, y)(New(y) \mid Server(x, y))$ then the depth of all reachable configurations is bounded by 2.

2.2 WQOs, BQOs, and WSTSs

We briefly recall the relevant theory of well-quasi-orderings, better-quasi-orderings [23], and well-structured transition systems [1, 9, 12].

Well-quasi-ordering. A pair $(X, \leq)$ of a set X and a binary relation $\leq$ on X is called *well-quasi-ordered set* (wqo) if and only if (1) $\leq$ is a quasi-ordering (i.e., reflexive and transitive) and (2) any infinite sequence $x_0, x_1, x_2, \ldots$ of elements from X contains an increasing pair $x_i \leq x_j$ with $i < j$. A nonempty set $Y \subseteq X$ is called *directed* if for any $x, y \in Y$ there exists $z \in Y$ with $x, y \leq z$. A set $Y \subseteq X$ is called *upward-closed* if for any pair x, y such that $x \in Y$ and $y \geq x$ implies $y \in Y$. Similarly, Y is called *downward-closed* if for any pair x, y such that $x \in Y$ and $y \leq x$ implies $y \in Y$. The upward-closure of $Y \subseteq X$ is defined as $\uparrow Y = \{\, x \mid \exists y \in Y.\, x \geq y \,\}$. Correspondingly, we denote by $\downarrow Y$ the downward-closure of Y. We extend the ordering $\leq$ to an ordering $\leq$ on subsets of X as expected: for $Y_1, Y_2 \subseteq X$, we have $Y_1 \leq Y_2$ iff for all $y_1 \in Y_1$ there exists $y_2 \in Y_2$ if $y_1 \leq y_2$. For $Y \subseteq X$ we call $Y' \subseteq X$ *large* in Y iff $Y \leq Y'$. Conversely, we call Y' *small* in Y if $Y' \leq Y$. A subset $Y \subseteq X$ of X is called *irreducible* if for any $Y_1, Y_2 \subseteq X$, $Y \leq Y_1 \cup Y_2$ implies $Y \leq Y_1$ or $Y \leq Y_2$.

Better-quasi-orderings. Let $\leq$ be a quasi-ordering on a set X then define the quasi-ordering $\leq_1$ on subsets of X as follows: for $Y_1, Y_2 \subseteq X$, we have $Y_1 \leq_1 Y_2$ iff there exists an injection $\phi : Y_1 \to Y_2$ such that for all $y_1 \in Y_1$, $y_1 \leq \phi(y_1)$. We are interested in wqo sets $(X, \leq)$ whose powerset is again a wqo with respect to $\leq_1$. For this purpose we consider Nash-William's better-quasi-orderings [23]. Better-quasi-ordered (bqo) sets are particular well-behaved wqo sets. Unlike general well-quasi-orderings, bqo sets are closed under the powerset construction. The formal definition of better-quasi-orderings is rather technical and not required for understanding this paper. We therefore refer to [23] for the actual definition. We only state the properties of bqo sets that we will use in our proofs.

Proposition 3. *Let $(X, \leq)$ be a bqo then*

1. *$(X, \leq)$ is a wqo,*
2. *$(2^X, \leq_1)$ is a bqo,*
3. *every $Y \subseteq X$ is a bqo with respect to the restriction of $\leq$ to Y.*

Properties 1 and 2 are proved in [23]. Property 3 immediately follows from the definition of bqo sets.

Well-structured transition system. A *well-structured transition systems* (WSTS) is a transition system $T = (S, s_0, \rightarrow, \leq)$ where S is a set of configurations, $s_0 \in S$ an initial configuration, $\rightarrow \subseteq S \times S$ a transition relation, and $\leq \subseteq S \times S$ a relation satisfying the following two conditions: (well-quasi-ordering) $\leq$ is a well-quasi-ordering on S; and (compatibility) $\leq$ is upward compatible with respect to $\rightarrow$, i.e., for all s_1, s_2, t_1 such that $s_1 \leq t_1$ and $s_1 \rightarrow s_2$, there exists t_2 such that $t_1 \rightarrow^* t_2$ and $s_2 \leq t_2$.

Definition 4 (Covering Problem). *Given a WSTS $(S, s_0, \rightarrow, \leq)$ and a configuration $t \in S$, the* covering problem *asks whether there exists a configuration $t' \in S$ such that $s_0 \rightarrow^* t'$ and $t \leq t'$.*

3 The Covering Problem for Depth-Bounded Processes

We define the following natural quasi-ordering $\leq$ on configurations of processes: let $P \equiv (\nu \boldsymbol{x})P'$ and Q be configurations then $P \leq Q$ if and only if $Q \equiv (\nu \boldsymbol{x})(P' \mid R)$ for some process term R. Meyer [18] proved that depth-bounded sets of configurations are well-quasi-ordered by $\leq$. Thus, a depth-bounded process $\mathcal{P} = (I, \mathcal{E})$ induces a well-structured transition systems $(Reach(\mathcal{P}), I, \rightarrow_E, \leq)$. We are interested in the covering problem for these WSTSs.

Forward vs. backward algorithms. The standard algorithm for deciding the covering problem for a WSTS is a backward algorithm that works as follows. Starting from the configuration t that is to be covered one computes the set of backward-reachable configurations of the upward closure of t and then checks whether this set contains the initial configuration. The well-quasi-ordering ensures that the backward analysis terminates.

In the WSTS $(Reach(\mathcal{P}), I, \rightarrow_E, \leq)$ that is induced by a depth-bounded process $\mathcal{P}$ we implicitly restrict the transition relation $\rightarrow_E$ to the forward-reachable configurations $Reach(\mathcal{P})$. The predecessor configurations with respect to this restricted transition relation are not effectively computable, i.e., the backward algorithm is not applicable to this WSTS. On the other hand, predecessor configurations for the unrestricted transition relation are effectively computable, but the induced set of backward-reachable configurations is in general not depth-bounded (and thus not well-quasi-ordered by $\leq$). A backward algorithm can only be effectively applied to the WSTS $(\mathcal{C}(k), I, \rightarrow_E, \leq)$. Here $\mathcal{C}(k)$ is the set of all configurations of depth k and k is the maximal depth of configurations in $Reach(\mathcal{P})$, i.e., $Reach(\mathcal{P}) \subseteq \mathcal{C}(k)$. Since k must be known in advance, Meyer's result only implies that the covering problem is decidable for depth-bounded

processes of known depth. We will show that there exists a forward algorithm that overcomes this limitation.

Besides the theoretical deficiency of backward algorithms there is also a practical reason why forward algorithms are more attractive. We explain this with an example.

Example 5. Consider the parameterized process $\mathcal{P}(n)$ for $n \in \mathbb{N}$ that is given by the initial configuration $I(n)$:

$$(\nu x, z, y, y_1, \ldots, y_n)(\mathit{Buffer}_n(x, z, y_1, \ldots, y_n) \mid \mathit{Env}(z, x, y))$$

and the equations $\mathcal{E}(n)$:

$$\begin{aligned}
\mathit{Buffer}_0(x, z) &= x(y).\mathit{Buffer}_1(x, z, y) \\
\mathit{Buffer}_i(x, z, y_1, \ldots, y_i) &= x(y).\mathit{Buffer}_{i+1}(x, z, y_1, \ldots, y_i, y) \\
&\quad + \overline{z}(y_1).\mathit{Buffer}_{i-1}(x, z, y_2, \ldots, y_i) \qquad \text{for } 0 < i < n \\
\mathit{Buffer}_n(x, z, y_1, \ldots, y_n) &= \overline{z}(y_1).\mathit{Buffer}_{n-1}(x, z, y_2, \ldots, y_n) \\
\mathit{Env}(z, x, y) &= \overline{x}(y).(\nu u)(\mathit{Env}(z, x, u)) \;+\; z(u).\mathit{Env}(z, x, u)
\end{aligned}$$

The process $\mathcal{P}(n)$ models a finite FIFO buffer that stores data sent by the environment in a queue of maximal length n. The queue is modeled using the parameter lists of the process identifiers Buffer_i.

Suppose we want to check that the configuration $P \equiv (\nu x, z)(\mathit{Buffer}_0(x, z))$ is coverable in $\mathcal{P}(n)$. The number of representatives for the set of configurations that are backward-reachable from the upward-closure of P grows exponential in n. The reason is that in one of the continuations of the choices that define $\mathit{Buffer}_i(x, z, y_1, \ldots, y_i)$ the parameter y_1 does not occur. A backward algorithm that computes the predecessors for the execution of this choice has no knowledge about the name that the parameter y_1 denotes. It has to guess whether it is a fresh name or whether it is equal to one of the other names appearing in the continuation. On the other hand, a forward algorithm always knows which name the parameter y_1 denotes. Therefore, the set of representatives for the configurations that are forward reachable from $I(n)$ grows only linear in n. It is this phenomenon that makes forward algorithms more appealing for the analysis of π-calculus processes.

4 An Adequate Domain of Limits

Most forward algorithms for solving the covering problem of WSTSs compute the *cover*, i.e., the downward-closure of the forward-reachable configurations and then check whether this set contains the configuration to be covered. In order to effectively compute the cover, one needs to find a *completion* of the wqo set that contains all the limits of downward-closed sets. The canonical example is the completion for the well-quasi-ordering on markings of Petri nets. It is given by vectors over the set $\mathbb{N}_\omega$ of natural numbers extended with the limit ordinal ω. This completion is the basis for the Karp-Miller algorithm [15] that computes the covering tree of a Petri net. The notion of an *adequate domain of limits* [10, 13] formalizes the completions of wqo sets.

An *adequate domain of limits* (ADL) [13] for a well-quasi-ordered set $(X, \leq)$ is a tuple $(Y, \sqsubseteq, \gamma)$ where Y is a set disjoint from X; (L1) the map $\gamma : Y \cup X \to 2^X$ is such that $\gamma(z)$ is downward-closed for all $z \in X \cup Y$, and $\gamma(x) = \downarrow \{x\}$ for all $x \in X$; (L2) there is a limit point $\top \in Y$ such that $\gamma(\top) = X$; (L3) $z \sqsubseteq z'$ if and only if $\gamma(z) \subseteq \gamma(z')$; and (L4) for any downward-closed set D of X, there is a finite subset $E \subseteq Y \cup X$ such that $\gamma(E) = D$, where γ is extended to sets as expected: $\gamma(E) = \bigcup_{z \in E} \gamma(z)$. A *weak adequate domain of limits* (WADL) [10] for $(X, \leq)$ is a tuple $(Y, \sqsubseteq, \gamma)$ satisfying (L1),(L3), and (L4). Note that any weak adequate domain of limits can be extended to an adequate domain of limits.

4.1 Limit Configurations

We now describe a weak adequate domain of limits for depth-bounded configurations. In order to finitely represent the limits of infinite downward-closed sets we need to be able to express that certain subterms in a configuration can be replicated arbitrarily often. A natural solution to this problem is to extend configurations with the replication operator ! that is used as a recursion primitive in alternative definitions of the π-calculus [21, 22]. Instead of using replication to express recursion, we use it to effectively represent infinite sets of configurations.

A *limit configuration* E is constructed recursively from process identifiers $A(\boldsymbol{x})$, parallel composition $E_1 \mid E_2$, name restriction $(\nu x)E$ and replication $!E$. We extend the congruence relation $\equiv$ from configurations to limit configurations by adding the axiom $!E \equiv (E \mid !E)$. We carry over the definitions of the transition relations of processes and the quasi-ordering $\leq$ from configurations to limit configurations by replacing the congruence relation in the definitions with the extended congruence relation. We then define the denotation $\gamma(E)$ of a limit configuration E as its downward-closure restricted to non-limit configurations:

$$\gamma(E) = \{\, P \mid P \text{ configuration and } P \leq E \,\}$$

The quasi-ordering $\sqsubseteq$ on limit configurations that is required for the adequate domain of limits is defined by condition (L3).

Example 6. Consider again the client-server process presented in Example 2. The following limit configuration denotes the cover of this process:

$$\begin{array}{l}(\nu x)((\nu y)(\mathit{New}(y) \mid \mathit{Server}(x,y)) \\ \quad \mid !(\nu z)(\mathit{Client}(z,x) \mid \mathit{Answer}(z)) \\ \quad \mid !(\nu z)(\mathit{Client}(z,x) \mid \mathit{Request}(x,z)))\end{array}$$

We now state the main technical result of this paper. Given a finite set of process identifiers PI, we denote by $\mathcal{C}(PI, k)$ the set of all configurations over PI that have depth at most k. We further denote by $\mathcal{L}(PI, k)$ the set of all limit configurations over PI whose elements denote sets of k-bounded configurations such that $\mathcal{L}(PI, k)$ itself does not contain the configurations in $\mathcal{C}(PI, k)$.

Theorem 7. *Let $k \in \mathbb{N}$ and let PI be a finite set of process identifiers. Then $(\mathcal{L}(PI, k), \sqsubseteq, \gamma)$ is a weak adequate domain of limits for the well-quasi-ordered set $(\mathcal{C}(PI, k), \leq)$.*

In the remainder of this section we prove Theorem 7.

4.2 Tree Encoding of Depth-Bounded Configurations

We first relate depth-bounded configurations with graphs of bounded tree-depth, which in turn can be encoded into trees of bounded height. The construction is similar to the one used in [18]. However, we prove that the tree encodings of depth-bounded configurations are not just well-quasi-ordered, but in fact better-quasi-ordered.

Communication topology. We use standard notation for (undirected) graphs. A *labelled graph* over a finite set of labels L is a tuple (G, l_v, l_e) where G is a graph, $l_v : V(G) \to L$ is a *vertex labelling function*, and $l_e : V(E) \to L$ is an *edge labelling function*.

Let $\mathcal{P} = (I, \mathcal{E})$ be a process. Let further n be the maximal arity of all vectors of names occurring in I and $\mathcal{E}$, and let $\mathcal{A}$ be the set of all process identifiers occurring in $I, \mathcal{E}$. Define the set of labels $L \stackrel{\text{def}}{=} 2^{\{0,\ldots,n\}} \cup \mathcal{A} \cup \{\bullet\}$ where $\bullet$ is distinct from all process identifiers. Let P be a configuration of process $\mathcal{P}$ of the form

$$(\nu \boldsymbol{x})(\Pi_{j \in J} A_j(\boldsymbol{x}_j))$$

where $\boldsymbol{x} = x_1, \ldots, x_m$, and the index sets $\{1..m\}$, and J are disjoint. The function ct maps P to a labelled graph over L as follows: the graph consists of vertices corresponding to threads and names occurring in the configuration. Each thread vertex is labelled by the process identifier of the corresponding thread in the configuration. There are edges between thread vertices and name vertices indicating that one of the names in the parameter vector of the thread is the name associated with that name vertex. Formally, $\mathsf{ct}(P)$ is a graph $((V, E), l_v, l_e)$ where

- V is a union of disjoint sets of vertices $\{v_j\}_{j \in J}$ and $\{v_1, \ldots, v_m\}$,
- $E = \{\, \{v_j, v_i\} \mid j \in J \ \wedge\ 1 \le i \le m \ \wedge\ x_{j_r} = x_i \text{ for some } 1 \le r \le n \,\}$,
- $l_v(v_k) = \begin{cases} A_k & \text{if } k \in J \\ \bullet & \text{otherwise,} \end{cases}$
- $l_e(\{v_j, v_i\}) = \{\, r \mid j \in J \ \wedge\ 1 \le i \le m \ \wedge\ x_{j_r} = x_i \,\}$.

We call $\mathsf{ct}(P)$ the *communication topology* of configuration P.

Tree-depth. We relate depth-bounded sets of configurations to sets of graphs of bounded tree-depth [24]. A *path* π in a graph G is a sequence $v_1, \ldots, v_n$ of vertices in $V(G)$ that are consecutively connected by edges in $E(G)$. We say that π *connects* vertices v_1 and v_n. We call π *simple path* if for all $1 \le i < j \le n$, $v_i \neq v_j$. A *tree* T is a graph such that every pair of distinct vertices in T is connected by exactly one path and this path is simple. A *rooted tree* is a tree with a dedicated root vertex. A *rooted forest* is a disjoint union of rooted trees. The *height* of a vertex v in a rooted forest F, denoted $\mathit{height}(F, v)$, is the number of vertices on the path from the root (of the tree to which v belongs) to v. The *height* of F is the maximal height of the vertices in F. Let v, w be vertices of F and let T be the tree in F to which w belongs. The vertex v is an *ancestor* of vertex w in F, denoted $v \preceq w$, if v belongs to the path connecting w and the root of T. The *closure* $\mathit{clos}(F)$ of a rooted forest F is the graph consisting of the vertices of F and the edge set $\{\, \{v, w\} \mid v \preceq w, v \neq w \,\}$. The *tree-depth* $\mathsf{td}(G)$ of a graph G is the minimum height of all rooted forests F such that $G \subseteq \mathit{clos}(F)$. The tree-depth

of a labelled graph is the tree-depth of the enclosed graph. Finally, we say that a set of graphs $\mathcal{G}$ has *bounded tree-depth* if there exists $k \in \mathbb{N}$ such that all graphs $G \in \mathcal{G}$ have tree-depth at most k.

Proposition 8. *A set of configurations $\mathcal{C}$ is depth-bounded iff its communication topologies $\mathsf{ct}(\mathcal{C})$ have bounded tree-depth.*

The proof of Proposition 8 uses Meyer's characterization of sets of depth-bounded configurations in terms of sets of graphs that are bounded in the length of the simple paths [18, Theorem 1]. One can easily show that a set of graphs is bounded in the length of the simple paths if and only if it has bounded tree-depth.

We now relate the ordering on configurations $P \leq Q$ with an ordering on the underlying communication topologies. Given two labelled graphs G_1 and G_2, we say G_1 is (isomorphic to) a *subgraph* of G_2, written $G_1 \hookrightarrow G_2$, iff there exists an injective label-preserving homomorphism from G_1 to G_2.

Lemma 9. *Let P and Q be configurations. Then $P \leq Q$ iff $\mathsf{ct}(P) \hookrightarrow \mathsf{ct}(Q)$.*

Tree encoding. A labelled rooted tree over a finite set of labels L is a pair (T, l) where T is a rooted tree and $l : V(T) \to L$ a vertex labelling function. We extend the relation $\hookrightarrow$ to rooted labelled trees, as expected, and we say that a tree T_1 is a *subtree* of tree T_2 whenever $T_1 \hookrightarrow T_2$ holds. In the following, we fix a finite set of labels L. Let L_k be the set of all isomorphism classes of labelled graphs G over labels $L \cup (L \times \{1..k\})$ such that G has at most k vertices. Clearly, since L is finite, L_k is finite.

Given a labelled graph G over labels L that has tree-depth at most k, we can construct a labelled rooted tree (T, l) over the set of labels L_k from G as follows. First, let F be a rooted forest of minimal height whose closure contains the graph induced by G. The rooted tree T is constructed from the forest F by extending F with a fresh root vertex r that has edges to all the roots of the trees in F. The labelling function l is defined as follows. Let $v \in V(T)$ be a vertex in T. If $v = r$ then $l(r)$ is the empty graph. Otherwise v is a vertex in F (and thus in G). Let P be the subgraph of G that is induced by the vertices on the path from v to the root (of the tree in F to which v belongs). Now construct a graph P_h from P by adding to the label of each vertex of P its height in F. Then $l(v)$ is the isomorphism class of P_h. Since G has tree-depth at most k, $P_h \in L_k$. Thus, l is well-defined. Let Trees_k be the function mapping a labelled graph G of tree-depth at most k to the set of all labelled rooted trees over L_k that can be constructed from G as described above. We denote by $\mathrm{rng}(\mathit{Trees}_k)$ the set of labelled trees $\bigcup \{\ \mathit{Trees}_k(G) \mid G$ labelled graph over L with $\mathsf{td}(G) \leq k\ \}$.

Lemma 10. *Let $k \in \mathbb{N}$ and T_1, T_2 be trees in $\mathrm{rng}(\mathit{Trees}_k)$. If T_1 is a subtree of T_2 then $G_1 \hookrightarrow G_2$ for all G_1, G_2 with $T_1 \in \mathit{Trees}_k(G_1)$ and $T_2 \in \mathit{Trees}_k(G_2)$.*

Let T be a rooted tree and $x, y \in V(T)$ two vertices. The infimum of x and y, denoted $x \inf y$, is the vertex $z \in V(T)$ with the greatest height such that $z \preceq x$ and $z \preceq y$. Given rooted trees T_1 and T_2, a function φ is an *inf-preserving embedding* from T_1 into T_2 iff (1) $\varphi : V(T_1) \to V(T_2)$ is injective, and (2) for all $x, y \in V(T_1)$, $\varphi(x \inf y) = \varphi(x) \inf \varphi(y)$. An embedding between two rooted labelled trees over the same set of labels is *label-preserving* iff it maps vertices to vertices with the same label.

Clearly, if a tree is a subtree of another tree then there exists an inf and label preserving embedding between these trees. For trees that result from the tree encoding of configurations the converse holds, too. Vertices of different height in such trees have always different labels. Thus, an inf and label-preserving embedding between such trees also preserves antecedence of vertices.

Lemma 11. *Let $k \in \mathbb{N}$ and T_1, T_2 be trees in $\mathrm{rng}(\mathit{Trees}_k)$. Then the following two properties are equivalent:*

1. *there exists an inf and label-preserving embedding from T_1 to T_2;*
2. *T_1 is a subtree of T_2.*

Laver [16] proved a variation of Kruskal's tree theorem for trees labelled by a bqo set, namely that countable rooted trees labelled by a bqo set are a bqo under inf-preserving embedding. Similar to Friedman's special case of Kruskal's tree theorem, we get the special case of Laver's theorem that rooted trees labelled by a finite set of labels are better-quasi-ordered by inf and label-preserving embedding. Thus, together with Lemma 10 we get the following proposition.

Proposition 12. *For any $k \in \mathbb{N}$, $(\mathrm{rng}(\mathit{Trees}_k), \hookrightarrow)$ is a bqo set.*

4.3 Limit Configurations as Ideal Completions

Finkel and Goubault-Larrecq [10] characterize the minimal candidates for the WADLs of a wqo set X in terms of its ideal completion. This means that the set of all downward-closed directed subsets of X forms a WADL for X. We use this observation to prove that limit configurations form WADLs for depth-bounded configurations.

Proposition 13. *The directed downward-closed sets of depth-bounded configurations are exactly the denotations of limit configurations.*

By [10, Proposition 3.3] the above proposition implies Theorem 7. In our proof of Proposition 13 we characterize the tree encodings of downward-closed sets of configurations in terms of the languages of *hedge automata* [5, Chapter 8].

Hedge automata. A *(nondeterministic) finite hedge automaton* $\mathcal{A}$ over a finite alphabet Σ is a tuple (Q, Σ, Q_f, Δ) where Q is a finite set of states, $Q_f \subseteq Q$ is a set of final states, and Δ is a finite set of transition rules of the following form:

$$a(R) \to q$$

where $a \in \Sigma$, $q \in Q$, and $R \subseteq Q^*$ is a regular language over Q. These languages R occuring in the transition rules are called *horizontal languages*.

A *run* of $\mathcal{A}$ on a rooted labelled tree T with vertex label function $l : V(T) \to \Sigma$ is a vertex label function $r : V(T) \to Q$ such that for each vertex $v \in V(T)$ with $a = l(v)$ and $q = r(v)$ there is a transition rule $a(R) \to q$ with $r(v_1) \dots r(v_n) \in R$ where $v_1, \dots, v_n$ are the immediate successors of v in T. In particular, to apply a rule to a leaf, the empty word ϵ has to be in the horizontal language of the rule R.

A rooted labelled tree T is *accepted* by $\mathcal{A}$ if there is a run r of $\mathcal{A}$ on T such that r labels the root of T by a final state. The *language* $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of all rooted labelled trees over Σ that are accepted by $\mathcal{A}$.

Finite partitions of well-quasi-ordered sets. In order to characterize the horizontal languages of the constructed hedge automaton we will define equivalence classes on the vertices of the individual levels of the tree encodings. For this purpose, the following definition will be useful. Let $(X, \leq)$ be a well-quasi-ordered set. We call a partition $\mathcal{P} \subseteq 2^X$ of X an *infinite chain partition* if and only if (1) $\mathcal{P}$ is finite and (2) for all $Y \in \mathcal{P}$, either Y is a singleton or Y contains an infinite chain C such that $Y \leq C$.

Proposition 14. *If $(X, \leq)$ is a countable well-quasi-ordered set then there exists an infinite chain partition of X.*

In order to prove Proposition 13, we first prove that every directed downward-closed set of depth-bounded configurations is the denotation of a limit configuration.

Lemma 15. *Let D be a downward-closed set of configurations then there exists a limit configuration E such that $D = \gamma(E)$.*

For proving the lemma, let $D = (P_i)_{i\in\mathbb{N}}$ be a downward-closed directed family of configurations and let k be the maximal tree-depth of the graphs in $\mathsf{ct}(D)$. Choose some $Q_0 \in D$ whose communication topology has tree-depth k. Using Q_0 construct an ascending chain $D' = (Q_i)_{i\in\mathbb{N}}$ as follows: for each $i \in \mathbb{N}$ choose $Q_i \in D$ such that $P_i \leq Q_i$ and $Q_{i-1} \leq Q_i$. Such Q_i exists for each $i \in \mathbb{N}$ since D is directed and, by induction, $Q_{i-1} \in D$. Then by construction (1) $D = \downarrow D'$ and (2) all elements in D' have tree-depth k. Let $(G_i)_{i\in\mathbb{N}}$ be the family of labelled graphs $G_i = \mathsf{ct}(Q_i)$. Now for each $i \in \mathbb{N}$ choose a tree $T_i \in \mathit{Trees}_k(G_i)$ such that the family $\mathcal{T} = (T_i)_{i\in\mathbb{N}}$ is an ascending chain with respect to the subtree relation. Such a family exists because the G_i are ordered by subgraph isomorphism and all G_i have the same tree-depth. Without loss of generality we assume that the vertex sets of all trees T_i are pairwise disjoint.

Let $V = \bigcup_{i\in\mathbb{N}} V(T_i)$, $E = \bigcup_{i\in\mathbb{N}} E(T_i)$, and let l be the union of all the vertex labelling functions of the labelled trees T_i. The height of the vertices in the trees T_i range from 1 to $k+1$. For a vertex $v \in V$ of height $h > 1$ we denote by $\mathit{parent}(v) \in V$ the parent of v in the tree T_i to which v belongs. Similarly, for a vertex $v \in V$ we denote by $\mathit{Children}(v)$ the set of all vertices that are direct successors of v in the tree to which v belongs. We extend the functions parent to sets of vertices, as expected. Furthermore, let $T(v)$ be the subtree rooted in v of the tree T_i with $v \in V(T_i)$. For all $1 \leq h \leq k+1$, let V_h be the set of all vertices in V that have height h. For all h we extend the relation $\hookrightarrow$ from labelled rooted trees to vertices in V_h as expected: for all $v, w \in V_h$, $v \hookrightarrow w$ iff $T(v) \hookrightarrow T(w)$. From Proposition 12 and Property 3 of Proposition 3 follows that for all h, $(V_h, \hookrightarrow)$ is a bqo.

We will now construct a finite hedge automaton $\mathcal{A}$ from the family of trees $\mathcal{T}$ whose language is both small and large in $\mathcal{T}$. For this purpose we define an equivalence relation on each V_h that partitions V_h into finitely many equivalence classes. These equivalence classes serve as the states of the automaton.

For each $i \in \mathbb{N}$ fix some injective label-preserving homomorphism $\phi_i : V(T_i) \to V(T_{i+1})$ and denote by $\phi_{[i,j]}$ the composition $\phi_{j-1} \circ \cdots \circ \phi_i$ if $j > i$ and the identify function id if $j = i$. Then define an equivalence relation $\sim$ on V as follows: for all $v_i \in V(T_i)$ and $v_j \in V(T_j)$

$$v_i \sim v_j \quad \text{iff} \quad \begin{array}{l} i \leq j \text{ and } \phi_{[i,j]}(v_i) = v_j \text{ or} \\ i \geq j \text{ and } \phi_{[j,i]}(v_j) = v_i \end{array}$$

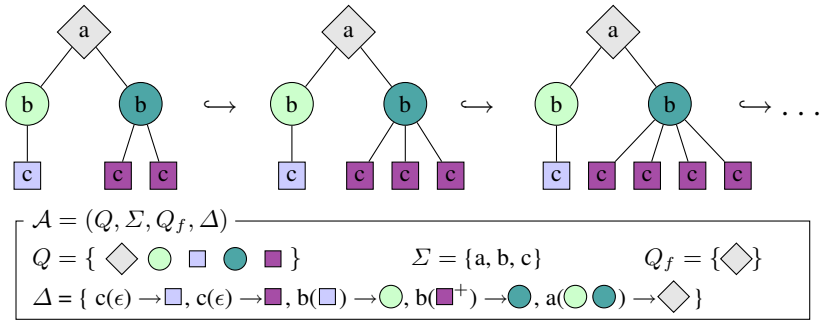

Fig. 1. A chain of labelled trees with the equivalence classes under the relations $\simeq_h$ and the constructed hedge automaton

Now, recursively define an equivalence relation $\simeq_h$ on V_h for each $1 \leq h \leq k+1$ as follows: for $h = 1$ we simply have $v \simeq_1 w$ for all $v, w \in V_1$. In order to define $\simeq_h$ for $h > 1$ we need some intermediate definitions. Given an equivalence class U in the quotient of V_{h-1} wrt. $\simeq_{h-1}$, let $\mathit{Children}(U)$ be the set of equivalence classes $\tilde{v}$ in the quotient $V_h/_\sim$ such that some $v \in \tilde{v}$ has a parent in U. Since $(V_h, \hookrightarrow)$ is a bqo, and $\mathit{Children}(U) \subseteq 2^{V_h}$, it follows from Proposition 3 that $(\mathit{Children}(U), \hookrightarrow_1)$ is also a bqo and thus a wqo. Furthermore, $\mathit{Children}(U)$ is countable. Thus, by Proposition 14 there exists an infinite chain partition of $\mathit{Children}(U)$. For each U, choose one such infinite chain partition $\mathcal{P}(U)$ of $\mathit{Children}(U)$. Then for $v, w \in V_h$ we define: $v \simeq_h w$ iff there exists $U \in V_{h-1}/_{\simeq_{h-1}}$ such that (1) $\mathit{parent}(v), \mathit{parent}(w) \in U$ and (2) there is $P \in \mathcal{P}(U)$ such that $v, w \in \bigcup P$.

We can easily prove by induction on h that each $\simeq_h$ is indeed an equivalence relation on V_h and that each $\simeq_h$ partitions V_h into finitely many equivalence classes. Furthermore, using the definition of infinite chain partitions one can easily prove the following properties.

Lemma 16. *Let $U \in V_h/_{\simeq_h}$ then*

1. *all $v \in U$ have the same label,*
2. *U is directed with respect to $\hookrightarrow$,*
3. *if $h = 1$ then U contains exactly the root vertices of all the trees T_i,*
4. *if $h > 1$ then $\mathit{parent}(U) \subseteq U'$ for some $U' \in V_{h-1}/_{\simeq_{h-1}}$ and*
   (a) *either all vertices in U' have at most one child in U or*
   (b) *every $v \in U$ is contained in a proper infinite chain $C \subseteq U$ and for every finite subset $V \subseteq U$ there exists $v' \in U'$ such that $V \hookrightarrow_1 \mathit{Children}(v') \cap U$.*

Now let $\simeq$ be the union of all the relations $\simeq_h$. Then $\simeq$ is an equivalence relation on V that partitions V into finitely many equivalence classes. For an equivalence class $U \in V/_\simeq$, let $C(U)$ be the set of all equivalence classes that contain children of vertices in U. Furthermore, let $l(U)$ be the unique label of all vertices in U, and let $m(U)$ denote 1 if every parent of a vertex $v \in U$ has at most one child in U and, otherwise, let $m(U)$ denote the symbol $+$. Now define the hedge automaton $\mathcal{A} = (Q, \Sigma, Q_f, \Delta)$ where:

- $Q = V/_{\simeq}$,
- $\Sigma = L_k$,
- $Q_f = V_1/_{\simeq}$,
- Δ consists of transition rules of the following form for each $U \in V/_{\simeq}$
  - $l(U)(U_1^{m(U_1)} \cdots U_n^{m(U_n)}) \to U$ if $C(U) = \{U_1, \ldots, U_n\}$
  - $l(U)(\epsilon) \to U$ if $C(U) = \emptyset$.

Figure 1 depicts a chain of trees and the constructed automaton $\mathcal{A}$. The equivalence classes in the quotient $V/\simeq$ are highlighted in the trees.

Using Lemma 16 we can now prove that the language accepted by $\mathcal{A}$ is both small and large in $\mathcal{T}$.

Lemma 17. *$\mathcal{L}(\mathcal{A})$ is small in $\mathcal{T}$, i.e., $\forall T \in \mathcal{L}(\mathcal{A})\, \exists i \in \mathbb{N} :\ T \hookrightarrow T_i$.*

Lemma 18. *$\mathcal{L}(\mathcal{A})$ is large in $\mathcal{T}$, i.e., $\forall i \in \mathbb{N}\, \exists T \in \mathcal{L}(\mathcal{A}) :\ T_i \hookrightarrow T$.*

Note that by construction of $\mathcal{A}$ the tree encoding operation can be reversed on the trees in $\mathcal{L}(\mathcal{A})$. Let $D_{\mathcal{A}}$ be the corresponding set of configurations. From Lemmas 17,18,10, and 9 follows that $D = {\downarrow}D' = {\downarrow}D_{\mathcal{A}}$. From $\mathcal{A}$ we can now easily construct a limit configuration E whose denotation is the downward closure of $D_{\mathcal{A}}$. It follows that $D = {\downarrow}D_{\mathcal{A}} = \gamma(E)$ which proves Lemma 15.

Lemma 19. *For any limit configuration E, $\gamma(E)$ is a downward-closed directed set.*

Clearly $\gamma(E)$ is downward-closed. For proving that $\gamma(E)$ is directed, we can again construct a hedge automaton $\mathcal{A}$ from E, such that the tree encoding operation can be reversed on all trees accepted by $\mathcal{A}$ and the downward-closure of the resulting configurations $D_{\mathcal{A}}$ coincides with $\gamma(E)$. Using a simple pumping argument for the language $\mathcal{L}(\mathcal{A})$ we can show that for every two trees $T_1, T_2 \in \mathcal{L}(\mathcal{A})$ there exists a tree $T \in \mathcal{L}(\mathcal{A})$ such that $T_1 \hookrightarrow T$ and $T_2 \hookrightarrow T$. It follows that $D_{\mathcal{A}}$ is directed and thus $\gamma(E)$.

5 Forward Analysis of Depth-Bounded Processes

The expand, enlarge, and check (EEC) algorithm of Geeraerts et al. [13] is a forward algorithm that decides the covering problem for *effective* WSTSs with appropriate adequate domain of limits.

A WSTS $(X, x_0, \to, \leq)$ and an adequate domain of limits $(Y, \sqsubseteq, \gamma)$ for the wqo $(X, \leq)$ are *effective* if the following conditions are satisfied: (E1) X and Y are recursively enumerable; (E2) for any $x_1, x_2 \in X$, one can decide whether $x_1 \to x_2$; (E3) for any $z \in X \cup Y$ and for any finite subset $Z \subseteq X \cup Y$, one can decide whether $Post(\gamma(z)) \subseteq \gamma(Z)$; and (E4) for any finite subsets $Z_1, Z_2 \subseteq X \cup Y$, one can decide whether $\gamma(Z_1) \subseteq \gamma(Z_2)$.

We argue that the WSTS induced by a depth-bounded process together with its WADL of limit configurations are effective. The conditions (E1) and (E2) are clearly satisfied. Also given a limit configuration z we can compute a finite set of limit configurations denoting $Post(\gamma(z))$. Note further that Proposition 13 implies that for any finite subsets $Z_1, Z_2 \subseteq \mathcal{L}(PI, k)$, $\gamma(Z_1) \subseteq \gamma(Z_2)$ holds if and only if for all $z_1 \in Z_1$ there

exists $z_2 \in Z_2$ such that $\gamma(z_1) \subseteq \gamma(z_2)$. The inclusion problem $\gamma(z_1) \subseteq \gamma(z_2)$ can be reduced to the language inclusion problem for deterministic hedge automata, which is decidable. For this purpose, one computes deterministic hedge automata from the finitely many tree encodings of the configurations of z_1 and z_2 and then checks whether the language of some automaton of z_1 is included in the language of some automaton of z_2. Thus conditions (E3) and (E4) are also satisfied.

Finally, let us explain why the EEC algorithm terminates on depth-bounded systems even if the bound of the system is not known a priori. The idea of the algorithm is to simultaneously enumerate two infinite increasing chains. The first chain $X_0 \subseteq X_1 \ldots$ is a sequence of finite subsets of X that contains all reachable configurations of the analyzed system. The second chain $Y_0 \subseteq Y_1 \subseteq \ldots$ is a sequence of finite subsets of Y that contains all limits Y. In each iteration i the algorithm computes an under and an over-approximation of the analyzed system for the current pair (X_i, Y_i) of elements in the chain. These approximations are such that the under-approximation is guaranteed to detect that t can be covered if X_i contains a path to a covering state. The over-approximation is guaranteed to detect that t cannot be covered if Y_i can express $\downarrow Post^*(\downarrow s_0)$ and this set does not cover t. The conditions on the chains ensure that one of the two conditions eventually holds for some $i \in \mathbb{N}$.

For deciding the covering problem of depth-bounded systems we can now simply enumerate the sets $\mathcal{C}(PI) = \bigcup_{k\in\mathbb{N}} \mathcal{C}(PI, k)$ and $\mathcal{L}(PI) = \bigcup_{k\in\mathbb{N}} \mathcal{L}(PI, k)$. Then in each iteration of the EEC algorithm the pair (X_i, Y_i) is contained in some limit domain $\mathcal{C}(PI, k), \mathcal{L}(PI, k))$ and the conditions on the chains for termination of the EEC algorithm are still satisfied.

Theorem 20. *The covering problem for depth-bounded processes is decidable.*

Complexity. Depth-bounded processes subsume Petri nets [19]. This implies an exponential space lower bound on the complexity of the covering problem for depth-bounded processes [17]. The exact complexity is open.

6 Conclusion

At the dawn of cloud computing and processors that put an enormous number of cores at disposal of the programmer, message passing concurrency is gaining more and more importance. Many typical use cases of message passing such as client-server and consumer-producer communication with an unbounded number of clients/producers, and master-worker load balancing, can be modeled by depth-bounded processes. The covering problem plays a central role in the automated verification of the correctness of message passing systems. We prepared the ground for a spectrum of forward algorithms that solve the covering problem for depth-bounded processes. The exploitation of our result for the development of practical forward algorithms for this class of systems is our primary goal for future research.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *LICS*, pages 313–321, 1996.

2. R. M. Amadio and C. Meyssonnier. On decidability of the control reachability problem in the asynchronous pi-calculus. *Nord. J. Comput.*, 9(1):70–101, 2002.
3. J. Bauer and R. Wilhelm. Static analysis of dynamic communication systems by partner abstraction. In *SAS*, pages 249–264, 2007.
4. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definitions in channel based calculi. In *ICALP*, pages 133–144, 2003.
5. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. Available from: `http://tata.gforge.inria.fr/`, 2008. release November, 18th 2008.
6. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.
7. M. Dam. Model checking mobile processes. In E. Best, editor, *CONCUR*, volume 715 of *Lecture Notes in Computer Science*, pages 22–36. Springer, 1993.
8. A. Finkel. A Generalization of the Procedure of Karp and Miller to Well Structured Transition Systems. In *ICALP*, pages 499–508, 1987.
9. A. Finkel. Reduction and covering of infinite reachability trees. *Inf. Comput.*, 89(2):144–179, 1990.
10. A. Finkel and J. Goubault-Larrecq. Forward Analysis for WSTS, Part I: Completions. In *STACS*, volume 09001 of *Dagstuhl Sem. Proc.*, pages 433–444, 2009.
11. A. Finkel and J. Goubault-Larrecq. Forward Analysis for WSTS, Part II: Complete WSTS. In *ICALP (2)*, pages 188–199, 2009.
12. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
13. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Expand, Enlarge and Check: New algorithms for the coverability problem of WSTS. *J. Comput. Syst. Sci.*, 72(1):180–203, 2006.
14. D. Janssens, M. Lens, and G. Rozenberg. Computation graphs for actor grammars. *J. Comput. Syst. Sci.*, 46(1):60–90, 1993.
15. R. M. Karp and R. E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.
16. R. Laver. On Fraïssé's Order Type Conjecture. *Ann. of Math.*, 93(1):89–111, 1971.
17. R. J. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.
18. R. Meyer. On boundedness in depth in the pi-calculus. In *IFIP TCS*, volume 273 of *IFIP*, pages 477–489. Springer, 2008.
19. R. Meyer and R. Gorrieri. On the relationship between pi-calculus and finite place/transition petri nets. In *CONCUR*, pages 463–480, 2009.
20. R. Milner. Flowgraphs and flow algebras. *J. ACM*, 26(4):794–818, 1979.
21. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I. *Inf. Comput.*, 100(1):1–40, 1992.
22. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, II. *Inf. Comput.*, 100(1):41–77, 1992.
23. C. S. J. A. Nash-Williams. On better-quasi-ordering transfinite sequences. *Proc. Camb. Phil. Soc.*, 64:273–290, 1968.
24. J. Nešetřil and P. O. de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006.
25. K. Ostrovský. *On Modelling and Analysing Concurrent Systems*. PhD thesis, Chalmers University of Technology and Gotebörg University, 2005.
26. D. Sangiorgi. pi-calculus, internal mobility, and agent-passing calculi. *Theor. Comput. Sci.*, 167(1&2):235–274, 1996.

A Additional Proofs

Proposition 14. *If $(X, \leq)$ is a countable well-quasi-ordered set then there exists an infinite chain partition of X.*

Proof. We can construct an infinite chain partition $\mathcal{P}$ of X recursively using the following procedure: according to [**?**, Theorem 5], X can be partitioned into finitely many irreducible subsets $Y_1, \ldots, Y_n$. By [?, Proposition 3], for each $1 \leq i \leq n$, Y_i contains a chain C_i with $Y_i \leq C_i$. For each $1 \leq i \leq n$, check if Y_i contains an infinite chain with this property. If it does then add Y_i to $\mathcal{P}$. Otherwise pick one finite chain C_i with $Y_i \leq C_i$. Since C_i is finite it contains a greatest element y_i. Then let $Z_i = \{\, y \in Y_i \mid y_i \leq y \,\}$ be the set of elements in Y_i that are equivalent to y_i wrt. the quasi-ordering $\leq$. Since Y_i contains no infinite chains that are large in Y_i, the set Z_i is finite. Then add all singletons $\{z\}$ with $z \in Z_i$ to $\mathcal{P}$ and recursively apply the above procedure on the well-quasi-ordered set $(Y_i - Z_i, \leq)$. Clearly, if this procedure terminates then the resulting set $\mathcal{P}$ is an infinite chain partition of X. Thus, assume that the procedure does not terminate. Then the algorithm constructs a strictly decreasing infinite sequence $Y_1 \supseteq Y_2 \supseteq \ldots$ of subsets of Y with $Y_i - Y_{i+1} > Y_{i+1} - Y_{i+2}$ for all $i \in \mathbb{N}$. Define $X_i = Y_i - Y_{i+1}$ then each X_i is nonempty, i.e., we can choose $x_i \in X_i$ for each $i \in \mathbb{N}$ such that we get an infinite descending chain $x_1 > x_2 > \ldots$ of elements in X. This contradicts the fact that $\leq$ is well-founded. □

Lemma 17. *$\mathcal{L}(\mathcal{A})$ is small in $\mathcal{T}$, i.e., $\forall T \in \mathcal{L}(\mathcal{A})\, \exists i \in \mathbb{N} :\ T \hookrightarrow T_i$.*

Proof. For proving the lemma, let T_U be a tree labelled by L_k and r a run of $\mathcal{A}$ on T_U. We show by induction on the height of T_U that if $r(w) = U$ for the root w of T_U then there exists $v \in U$ such that $T_U \hookrightarrow T(v)$.

If $h = 1$ then T_U consists of a single root vertex w that is a leaf. Then the transition rule in Δ used to label w in r is of the form $l(U)(\epsilon) \to U$. Thus, by construction of $\mathcal{A}$ all trees $T(v)$ for vertices $v \in U$ consist of the single leaf vertex v labeled by $l(U)$, i.e., $T_U \hookrightarrow T(v)$ for all $v \in U$.

If $h > 1$ then the transition rule in Δ used to label w must have the form

$$l(U)(U_1^{m_1} \cdots U_n^{m_n}) \to U$$

with $C(U) = \{U_1, \ldots, U_n\}$ and $m_i = m(U_i)$ for all $1 \leq i \leq n$. Let

$$T_{1,1}, \ldots, T_{1,r_1}, \ldots, T_{n,1}, \ldots, T_{n,r_n}$$

be the subtrees of T_U rooted at the children of w such that r labels the root of each tree $T_{i,j}$ by U_i. These trees have height $h - 1$ and r is a run of $\mathcal{A}$ on each of these trees. Thus, by induction hypothesis there exist vertices

$$v_{1,1}, \ldots, v_{1,r_1} \in U_1 \ \ldots\ v_{n,1}, \ldots, v_{n,r_n} \in U_n$$

with $T_{i,j} \hookrightarrow T(v_{i,j})$ for all $1 \leq i \leq n$, $1 \leq j \leq r_i$. If two vertices $v_{i,j}$ and $v_{i',j'}$ coincide then we must have $i = i'$. Thus, $r_i > 1$ and $m(U_i) = +$, i.e., by construction

of $\mathcal{A}$, there are vertices in U that have more than one child in U_i. Then U_i satisfies property 4.(b) of Lemma 16, i.e., U_i contains a proper infinite chain C with $v_{i,j} \in C$. Hence, we can choose two vertices $v'_{i,j}, v'_{i',j'} \in C$ that are (1) distinct, (2) disjoint from all other $v_{i,j}$, and (3) satisfy $T_{i,j} \hookrightarrow T(v'_{i,j})$ and $T_{i,j} \hookrightarrow T(v'_{i',j'})$. Therefore, without loss of generality assume that all the $v_{i,j}$ are distinct. Now for any $1 \leq i \leq n$ we can find $v_i \in U$ such that

$$\{v_{i,1}, \ldots, v_{i,r_i}\} \hookrightarrow_1 \mathit{Children}(v_i) \cap U_i$$

Namely, if $r_i = 1$ then $v_i = \mathit{parent}(v_{i,1})$ and if $r_i > 1$ then such v_i exists by property 4.(b). Now, using the fact that U is directed we can inductively construct an upper bound $v \in U$ of all the v_i with respect to the wqo $\hookrightarrow$. Then we have by construction:

$$\{v_{1,1}, \ldots, v_{1,r_1}, \ldots, v_{n,1}, \ldots, v_{n,r_n}\} \hookrightarrow_1 \mathit{Children}(v)$$

We conclude that $\mathit{Children}(w) \hookrightarrow_1 \mathit{Children}(v)$ and $l(v) = l(U)$, i.e., $T_U \hookrightarrow T(v)$, which concludes the induction proof. □