Expand, Enlarge and Check... Made Efficient*

Gilles Geeraerts, Jean-François Raskin, and Laurent Van Begin**

Université Libre de Bruxelles – Département d'Informatique – CPI 212
Boulevard du Triomphe, B–1050 Bruxelles, Belgium
{gigeerae, jraskin, lvbegin}@ulb.ac.be

Abstract. The coverability problem is decidable for the class of well-structured transition systems. Until recently, the only known algorithm to solve this problem was based on symbolic backward reachability. In a recent paper, we have introduced the theory underlying a new algorithmic solution, called 'Expand, Enlarge and Check', which can be implemented in a forward manner. In this paper, we provide additional concepts and algorithms to turn this theory into efficient forward algorithms for monotonic extensions of Petri nets and Lossy Channels Systems. We have implemented a prototype and applied it on a large set of examples. This prototype outperforms a previous fine tuned prototype based on backward symbolic exploration and shows the practical interest of our new algorithmic solution.

1 Introduction

Model-checking is nowadays widely accepted as a powerful technique for the automatic verification of reactive systems that have natural finite state abstractions. However, many reactive systems are only naturally modelled as infinite-state systems. Consequently, a large (and successful) research effort has recently focused on the application of model-checking techniques to infinite-state models such as FIFO channel systems [2], (extensions of) Petri nets and automata with counters [14], broadcast protocols [7], etc.

One of the positive results is the decidability of the *coverability problem* for *well-structured transition systems* (WSTS for short). WSTS enjoy an infinite set of states that is well-quasi ordered by $\leq$ and their transition relation is monotonic w.r.t $\leq$. Examples of such systems are Petri nets and their monotonic extensions [4, 14], broadcast protocols [6], lossy channel systems [2]. The *coverability problem* asks whether a given WSTS S can reach a state of a given $\leq$-upward closed set of states U.

A general algorithm (i.e. a procedure that always terminates) is known to solve the coverability problem for WSTS [1, 9]. It symbolically manipulates upward-closed sets of states, obtained by unrolling the transition relation in a

* This work has been partially supported by the FRFC grant 2.4530.02. of the National Belgian Fund for Scientific Research (FNRS).
** Supported by a "First Europe" grant EPH3310300R0012 of the Walloon Region.

K. Etessami and S.K. Rajamani (Eds.): CAV 2005, LNCS 3576, pp. 394–407, 2005.

backward fashion. Unfortunately, backward search is seldom efficient in practice [12], and the only complete forward approach known so far is the Karp-Miller algorithm that can only be applied to a small subclass of WSTS: Petri nets. All the previous attempts to generalize this procedure have led to incomplete forward approaches that are either not guaranteed to terminate (e.g.: [6], as shown in [7]) or that can be inconclusive due to over-approximation [3].

Nevertheless, we have recently proposed a new schema of algorithms, called 'Expand, Enlarge and Check' (EEC for short), to solve the coverability problem for a large class of WSTS (those that enjoy reasonable effectiveness requirements, see [11] for the details). EEC works basically as follows. It constructs a sequence of pairs of approximations of the set of reachable states: an under-approximation (built during the 'Expand phase') and an over-approximation (built during the 'Enlarge' phase). Some basic results from the theory of well-quasi ordering and recursively enumerable sets allow us to show that positive instances of the coverability problem are answered by the sequence of under-approximations while negative instances are answered by the over-approximations after a finite number of iterations. The theory and the proofs are very elegant and furthermore the schema is really promising from the practical point of view because it can be implemented in a forward manner.

In this paper, we show that, indeed, EEC can be turned into an efficient algorithm to solve the coverability problem in a forward manner. In particular, we show how to implement the EEC efficiently for the two most practically important classes of WSTS in the literature: monotonic extensions of Petri Nets (EPN for short) and for Lossy Channel Systems (LCS for short). Those two classes are useful for the analysis of parametric systems and communication protocols. To obtain efficient algorithms from the EEC schema, we have to get over two obstacles: first, during the 'Expand' phase, we have to analyze finite graphs that can be very large. Second, during the 'Enlarge' phase, we have to approximate sets of successors efficiently. To solve the first problem, we show that we can always turn a WSTS into a *lossy* WSTS that respects the same coverability properties and for which the graph during the 'Expand' phase is monotonic. The coverability problem can often be solved efficiently in monotonic graphs because (roughly) only $\leq$-maximal states of the graph must be explored. We provide an efficient algorithm for that exploration. This algorithm is also applicable during the 'Enlarge' phase in the case of EPN. We show in the sequel that it dramatically improves the practical efficiency of the method. The second problem is difficult for LCS only. We provide here a way to construct efficiently the most precise approximations of the set of the successors of a downward-closed set of LCS configurations.

On the basis of those two conceptual tools, we have implemented a prototype to analyze coverability properties of EPN and LCS. We have applied the prototype to a large set of examples taken in the literature and compared its performances with our fine-tuned implementation of the backward search in the case of EPN. For LCS, the only available tools are implementing either a potentially non-terminating analysis or an over-approximation algorithm that is not

guaranteed to conclude due to over-approximation. The performance of our prototype are very encouraging and often much better than those of the backward search prototype.

The rest of the paper is organized as follows. In Section 2, we recall some basic notions about well-quasi orderings, WSTS and the coverability problem. In Section 3, we summarize the results of our previous paper about the EEC schema. In Section 4, we show how to efficiently explore monotonic graphs to establish coverability properties and show that the graphs that we have to analyze during the 'Expand' phase are monotonic when the lossy abstraction is applied. In Section 5, we show how EPN and LCS can be analyzed efficiently with the EEC schema, and provide practical evidence that it can compete advantageously with other techniques. Finally, we draw some conclusions in section 6.

Due to the lack of space, the proofs have been omitted in the present version of the paper. A full version of the paper and complete experimental results (including the data) can be found at: `http://www.ulb.ac.be/di/ssd/ggeeraer/eec/`

2 Preliminaries

In this section, we recall some fundamental results about *well-quasi orderings* and *well-structured transition systems* (the systems we analyze here). We show how to *finitely* represent upward- and downward-closed sets of states (which will allow us to devise *symbolic* algorithms), and discuss And-Or graphs and monotonic graphs (useful to represent abstractions of systems).

Well Quasi-Orderings and Adequate Domains of Limits. A *well quasi ordering* $\leq$ on C (wqo for short) is a *reflexive* and *transitive* relation s. t. for any infinite sequence $c_0 c_1 \dots c_n \dots$ of elements in C, there exist i and j, with $i < j$ and $c_i \leq c_j$. We note $c_i < c_j$ if $c_i \leq c_j$ but $c_j \not\leq c_i$.

Let $\langle C, \leq \rangle$ be a well-quasi ordered set. A $\leq$-*upward closed set* $U \subseteq C$ is such that for any $c \in U$, for any $c' \in C$ such that $c \leq c'$, $c' \in U$. A $\leq$-*downward-closed set* $D \subseteq C$ is such that for any $c \in D$, for any $c' \in C$ such that $c' \leq c$, $c' \in D$. The set of $\leq$-*minimal* elements $\mathsf{Min}(U)$ of a set $U \subseteq C$ is a minimal set such that $\mathsf{Min}(U) \subseteq U$ and $\forall s' \in U : \exists s \in \mathsf{Min}(U) : s \leq s'$. The next proposition is a consequence of wqo:

Proposition 1. *Let $\langle C, \leq \rangle$ be a wqo set and $U \subseteq C$ be an $\leq$-upward closed set, then: $\mathsf{Min}(U)$ is finite and $U = \{c \mid \exists c' \in \mathsf{Min}(U) : c' \leq c\}$.*

Thus, any $\leq$-upward closed set can be *effectively represented* by its finite set of minimal elements. To obtain a finite representation of $\leq$-downward-closed sets, we must use well-chosen limit elements $\ell \notin C$ that represent $\leq$-downward closures of infinite increasing chains of elements.

Definition 1 ([11]). *Let $\langle C, \leq \rangle$ be a well-quasi ordered set and L be a set s.t. $L \cap C = \emptyset$. The tuple $\langle L, \sqsubseteq, \gamma \rangle$ is called an* adequate domain of limits *for $\langle C, \leq \rangle$ if the following conditions are satisfied: (L_1: representation mapping) $\gamma : L \cup C \rightarrow$*

2^C associates to each element in $L \cup C$ a $\leq$-downward-closed set $D \subseteq C$, and for any $c \in C$, $\gamma(c) = \{c' \in C \mid c' \leq c\}$. γ is extended to sets $\mathcal{S} \subseteq L \cup C$ as follows: $\gamma(\mathcal{S}) = \cup_{c \in \mathcal{S}} \gamma(c)$; ($\mathsf{L}_2$: top element) There exists a special element $\top \in L$ such that $\gamma(\top) = C$; (L_3: precision order) The set $L \cup C$ is partially ordered by $\sqsubseteq$, where: $d_1 \sqsubseteq d_2$ iff $\gamma(d_1) \subseteq \gamma(d_2)$; ($\mathsf{L}_4$: completeness) for any $\leq$-downward-closed set $D \subseteq C$, there exists a finite set $D' \subseteq L \cup C$: $\gamma(D') = D$.

Well-Structured Transition Systems and Coverability Problem. A *transition system* is a tuple $S = \langle C, c_0, \rightarrow \rangle$ where C is a (possibly infinite) set of states, $c_0 \in C$ is the initial state, $\rightarrow \subseteq C \times C$ is a transition relation. We note $c \rightarrow c'$ for $\langle c, c' \rangle \in \rightarrow$. For any state c, $\mathsf{Post}(c)$ denotes the set of one-step successors of c, i.e. $\mathsf{Post}(c) = \{c' | c \rightarrow c'\}$, this function is extended to sets: for any $C' \subseteq C$, $\mathsf{Post}(C') = \bigcup_{c \in C'} \mathsf{Post}(c)$. Without loss of generality, we assume that $\mathsf{Post}(c) \neq \emptyset$ for any $c \in C$. A *path* of S is a sequence of states $c_1, c_2, \ldots, c_k$ such that $c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_k$. A state c' is reachable from a state c, noted $c \rightarrow^* c'$, if there exists a path $c_1, c_2, \ldots, c_k$ in S with $c_1 = c$ and $c_k = c'$. Given a transition system $S = \langle C, c_0, \rightarrow \rangle$, $\mathsf{Reach}(S)$ denotes the set $\{c \in C \mid c_0 \rightarrow^* c\}$.

Definition 2. *A transition system $S = \langle C, c_0, \rightarrow \rangle$ is a* well-structured transition system *(WSTS) [1, 9] for the quasi order $\leq \subseteq C \times C$ (noted: $S = \langle C, c_0, \rightarrow, \leq \rangle$) if: ($\mathsf{W}_1$: well-ordering) $\leq$ is a well-quasi ordering and (W_2: monotonicity) for any $c_1, c_2, c_3 \in C$: $c_1 \leq c_2$ and $c_1 \rightarrow c_3$ implies $\exists c_4 \in C : c_3 \leq c_4$ and $c_2 \rightarrow c_4$. It is* lossy *if its transition relation satisfies the following additional property: (W_3) $\forall c_1, c_2, c_3 \in C$ such that $c_1 \rightarrow c_2$ and $c_3 \leq c_2$, we have $c_1 \rightarrow c_3$.*

Problem 1. The *coverability problem for well-structured transition systems* is defined as follows: 'Given a well-structured transition system S and the $\leq$-upward closed set $U \subseteq C$, determine whether $\mathsf{Reach}(S) \cap U \neq \emptyset$'

To solve the coverability problem, we use the notion of covering set:

Definition 3. *For any WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$, the* covering set *of S ($\mathsf{Cover}(S)$), is the $\leq$-downward closure of $\mathsf{Reach}(S)$: $\mathsf{Cover}(S) = \{c | \exists c' \in \mathsf{Reach}(S) : c \leq c'\}$.*

Proposition 2 ([14]). *For any WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$, $\mathsf{Cover}(S)$ is such that for any $\leq$-upward closed set $U \subseteq C$: $\mathsf{Reach}(S) \cap U = \emptyset$ iff $\mathsf{Cover}(S) \cap U = \emptyset$.*

Finite Representation. For any WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$ with an adequate domain of limits $\langle L, \sqsubseteq, \gamma \rangle$ for $\langle C, \leq \rangle$, by property L_4 of Definition 1, there exists a finite subset $\mathsf{CS}(S) \subseteq L \cup C$ s.t. $\gamma(\mathsf{CS}(S)) = \mathsf{Cover}(S)$. In the sequel, $\mathsf{CS}(S)$ is called a *coverability set* of the covering set $\mathsf{Cover}(S)$ and finitely represents that set.

Lossiness Abstraction. Any WSTS can be turned into a lossy WSTS that respects the same coverability property. Definition 4 and Proposition 3 formalize this:

Definition 4. *The lossy version of a* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$ *is the lossy* WSTS $\mathsf{lossy}(S) = \langle C, c_0, \rightarrow_\ell, \leq \rangle$ *where* $\rightarrow_\ell = \{(c, c') \mid \exists c'' \in C : c \rightarrow c'' \wedge c' \leq c''\}$.

Proposition 3. *For any* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$*:* $\mathsf{Cover}(S) = \mathsf{Cover}(\mathsf{lossy}(S))$*; and for any* $\leq$*-upward closed set* $U \subseteq C$*:* $\mathsf{Reach}(S) \cap U = \emptyset$ *iff* $\mathsf{Cover}(\mathsf{lossy}(S)) \cap U = \emptyset$.

Abstractions. The EEC algorithm considers two kinds of abstractions. To represent them, we introduce two types of graphs. First, a $\overline{\preccurlyeq}$*-monotonic graph* is a finite graph $\langle V, \Rightarrow, v_i \rangle$ where V is a set of nodes, $\Rightarrow \subseteq V \times V$ is the transition relation and $v_i \in V$ is the initial node. That graph is associated with an order $\overline{\preccurlyeq} \subseteq V \times V$ such that for any $v_1, v_2, v_3 \in V$ with $v_1 \Rightarrow v_2$ and $v_1 \overline{\preccurlyeq} v_3$, there exists $v_4 \in V$ with $v_2 \overline{\preccurlyeq} v_4$ and $v_3 \Rightarrow v_4$. Notice that $\overline{\preccurlyeq}$-monotonic graphs are finite WSTS. Second, an *And-Or graph* is a tuple $G = \langle V_A, V_O, v_i, \Rightarrow \rangle$ where $V = V_A \cup V_O$ is the (finite) set of nodes (V_A is the set of "And" nodes and V_O is the set of "Or" nodes), $V_A \cap V_O = \emptyset$, $v_i \in V_O$ is the initial node, and $\Rightarrow \subseteq (V_A \times V_O) \cup (V_O \times V_A)$ is the transition relation s.t. $\forall v \in V_A \cup V_O$, there exists $v' \in V_A \cup V_O$ with $(v, v') \in \Rightarrow$.

Definition 5. *A* compatible unfolding *of an And-Or graph* $G = \langle V_A, V_O, v_i, \Rightarrow \rangle$ *is an infinite labelled tree* $T_G = \langle N, root, B, \Lambda \rangle$ *where: (i)* N *is the set of nodes of* T_G*, (ii)* $root \in N$ *is the root of* T_G*, (iii)* $B \subseteq N \times N$ *is the transition relation of* T_G*, (iv)* $\Lambda : N \rightarrow V_A \cup V_0$ *is the labelling function of the nodes of* T_G *by nodes of* G *that respects the three following compatibility conditions (*Λ *is extended to sets of nodes in the usual way):* $(\mathsf{C}_1) \Lambda(root) = v_i$*;* (C_2) *for all* $n \in N$ *such that* $\Lambda(n) \in V_A$*, we have that (a) for all nodes* $v' \in V_O$ *such that* $\Lambda(n) \Rightarrow v'$*, there exists one and only one* $n' \in N$ *such that* $B(n, n')$ *and* $\Lambda(n') = v'$*, and conversely (b) for all nodes* $n' \in N$ *such that* $B(n, n')$*, we have* $\Lambda(n) \Rightarrow \Lambda(n')$*.* (C_3) *for all* $n \in N$ *such that* $\Lambda(n) \in V_O$*, we have that: there exists one and only one* $n' \in N$ *such that* $B(n, n')$*, and* $\Lambda(n) \Rightarrow \Lambda(n')$*.*

Problem 2. The *And-Or Graph Avoidability Problem* is defined as: 'Given an And-Or graph $G = \langle V_A, V_O, v_i, \Rightarrow \rangle$ and $E \subseteq V_A \cup V_O$, is there $T = \langle N, root, B, \Lambda \rangle$, a compatible unfolding of G, s.t. $\Lambda(N) \cap E = \emptyset$?' When it is the case, we say that E is *avoidable* in G. It is well-known that this problem is complete for *PTIME*.

3 Expand, Enlarge and Check

This section recalls the fundamentals of the EEC algorithm [11]. The principle of this algorithm consists to build a sequence of pairs of approximations. The first one, is an under-approximation of the set of reachable states, and allows one to decide positive instances of the coverability problem. The latter one over-approximates the reachable states and is suitable to decide negative instances.

More precisely, given a WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$, and a set of limits L, the algorithm considers in parallel a sequence of subsets of C: $C_0, C_1, \ldots$ and a sequence of limit elements: $L_0, L_1, \ldots$ s.t. (i) $\forall i \geq 0 : C_i \subseteq C_{i+1}$, (ii) $\forall c \in \mathsf{Reach}(S) : \exists i \geq 0 : c \in C_i$, and (iii) $c_0 \in C_0$; and (iv) $\forall i \geq 0 : L_i \subseteq L_{i+1}$, (v)

$\forall \ell \in L : \exists i \geq 0 : \ell \in L_i$ and (vi) $\top \in L_0$. Given a set C_i, one can construct an *exact partial reachability graph* (EPRG for short) $\mathsf{EPRG}(S, C_i)$ which is an under-approximation of the system. Similarly, given a set L_i, one builds an over-approximation of the system under the form of an And-Or Graph:

Definition 6. *Given a* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$ *and a set* $C' \subseteq C$*, the* EPRG *of* S *is the transition system* $\mathsf{EPRG}(S, C') = \big\langle C', c_0, \big(\rightarrow \cap (C' \times C') \big) \big\rangle$.

Definition 7. *Given a* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$*, an adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$ *for* $\langle C, \leq \rangle$*, a finite subset* $C' \subseteq C$ *with* $c_0 \in C'$*, and a finite subset* $L' \subseteq L$ *with* $\top \in L'$*, the And-Or graph* $G = \langle V_A, V_O, v_i, \Rightarrow \rangle$*, noted* $\mathsf{Abs}(S, C', L')$*, is s.t.:* (A_1) $V_O = L' \cup C'$*;* (A_2) $V_A = \{\mathcal{S} \in 2^{L' \cup C'} \setminus \{\emptyset\} \mid \nexists d_1 \neq d_2 \in \mathcal{S} : d_1 \sqsubseteq d_2\}$*;* (A_3) $v_i = c_0$*;* $(\mathsf{A}_{4.1})$ *for any* $n_1 \in V_A, n_2 \in V_O$*:* $(n_1, n_2) \in \Rightarrow$ *if and only if* $n_2 \in n_1$*;* $(\mathsf{A}_{4.2})$ *for any* $n_1 \in V_O, n_2 \in V_A$: $(n_1, n_2) \in \Rightarrow$ *if and only if (i)* $\mathsf{Post}(\gamma(n_1)) \subseteq \gamma(n_2)$ *and (ii)* $\neg \exists n \in V_A : \mathsf{Post}(\gamma(n_1)) \subseteq \gamma(n) \subset \gamma(n_2)$.

Remark 1. The over-approximations are And-Or graphs instead of plain graphs. The reason is that, in Definition 1, we do not impose strong properties on the limits. Hence, the set of successors of an element $c \in L' \cup C'$ may not have a (unique) most precise approximation as a subset of $L' \cup C'$. Every over-approximation is able to simulate the successors in the WSTS, but some of them may lead to bad states, while others don't (which will be established by the And-Or graph).

Theorem 1 states the adequacy of these abstractions. Theorem 2 tells us that we will eventually find the right abstractions to decide the coverability problem. The EEC algorithm directly follows: it enumerates the pairs of C_i and L_i, and, for each of them, (1 – 'Expand') builds $\mathsf{EPRG}(S, C_i)$, (2 – 'Enlarge') builds $\mathsf{Abs}(S, C_i, L_i)$ and (3 – 'Check') looks for an error trace in $\mathsf{EPRG}(S, C_i)$ and checks the avoidability of the bad states in $\mathsf{Abs}(S, C_i, L_i)$ (see [11] for details).

Theorem 1. *Given a* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$ *with adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$*, and an* $\leq$*-upward-closed set* $U \subseteq C$*:* $\forall i \geq 0$*:* **If** $\mathsf{Reach}(\mathsf{EPRG}(S, C_i)) \cap U \neq \emptyset$ **then** $\mathsf{Reach}(S) \cap U \neq \emptyset$*.* **If** U *is avoidable in* $\mathsf{Abs}(S, C_i, L_i)$*,* **then** $\mathsf{Reach}(S) \cap U = \emptyset$.

Theorem 2. *Given a* WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$ *with adequate domain of limits* $\langle L, \sqsubseteq, \gamma \rangle$*, and an* $\leq$*-upward-closed set* $U \subseteq C$*:* **If** $\mathsf{Reach}(S) \cap U \neq \emptyset$*,* **then** $\exists i \geq 0 : \mathsf{Reach}(\mathsf{EPRG}(S, C_i)) \cap U \neq \emptyset$*.* **If** $\mathsf{Reach}(S) \cap U = \emptyset$*,* **then** $\exists i \geq 0$ *s.t.* U *is avoidable in* $\mathsf{Abs}(S, C_i, L_i)$.

Propositions 4 and 5 give properties of these abstractions. The latter says that, if C' is $\leq$-downward-closed and S is lossy, $\mathsf{EPRG}(S, C')$ is a finite $\leq$-monotonic graph. Section 4 shows how to efficiently explore these graphs.

Proposition 4. *Let* $S = \langle C, c_0, \rightarrow, \leq \rangle$ *be a* WSTS *and* $\langle L, \sqsubseteq, \gamma \rangle$ *be an adequate domain of limits, for* S*. Let* $\mathsf{Abs}(S, C', L') = \langle V_A, V_O, v_i, \Rightarrow \rangle$ *be an And-Or graph for some* $C' \subseteq C$ *and* $L' \subseteq L$*. For any* $v_1, v_2, v_3 \in V_A \cup V_O$ *s.t.* $v_1 \Rightarrow v_2$ *and* $\gamma(v_1) \subseteq \gamma(v_3)$*, there exists* $v_4 \in V_A \cup V_O$ *s.t.* $v_3 \Rightarrow v_4$ *and* $\gamma(v_2) \subseteq \gamma(v_4)$.

Proposition 5. *Given a* lossy WSTS $S = \langle C, c_0, \rightarrow, \leq \rangle$*, and a* $\leq$-downward-closed set $C' \subseteq C$*:* $\mathsf{EPRG}(S, C')$ *is a* $\leq$*-monotonic graph.*

4 Efficient Exploration of $\overline{\preccurlyeq}$-Monotonic Graphs

This section is devoted to the presentation of Algorithm 1 which efficiently decides the coverability problem on $\overline{\preccurlyeq}$-monotonic graphs. It is based on ideas borrowed from the algorithm to compute the minimal coverability set of Petri nets, presented in [8]. However, as recently pointed out in [10], this latter algorithm is flawed and may in certain cases compute an under-approximation of the actual minimal coverability set. Algorithm 1 corrects this bug in the context of finite graphs. At the end of the section, we show how to exploit it in EEC.

Algorithm 1 receives a $\overline{\preccurlyeq}$-monotonic graph $\mathcal{G} = \langle V, \Rightarrow, v_i \rangle$ and constructs a finite tree $\mathcal{T} = \langle N, n_0, B, \Lambda \rangle$, with set of nodes N, root node $n_0 \in N$, set of arcs $B \subseteq N \times N$ (in the following we denote by B^* the transitive closure of B) and labelling function $\Lambda : N \mapsto V$. We denote by $\mathsf{leaves}(\mathcal{T})$ the set of leaves of $\mathcal{T}$; by $\mathsf{subtree}(n)$, the maximal sub-tree of $\mathcal{T}$ rooted in $n \in N$; and by $\mathsf{nodes}(\mathcal{T})$ the set of nodes of $\mathcal{T}$. Given a tree $\mathcal{T}$, and two nodes n and n', the function $\mathsf{Replace\_subtree}(n, n', \mathcal{T}, to\_treat)$ replaces, in $\mathcal{T}$, the subtree $\mathsf{subtree}(n)$ by $\mathsf{subtree}(n')$ and removes from to_treat the nodes $n'' \in (\mathsf{leaves}(\mathsf{subtree}(n)) \setminus \mathsf{leaves}(\mathsf{subtree}(n'))) \cap to\_treat$. We also attach a predicate $\mathsf{removed}(n)$ to each node n of a tree, which is initially *false* for any node. When $\mathsf{removed}(n)$ is true, the node n is *virtually* removed from the tree. It is however kept in memory, so that it can be later put back in the tree (this makes sense since the bug in [8]

Data : $\mathcal{G} = \langle V, \Rightarrow, v_i \rangle$: $\overline{\preccurlyeq}$-monotonic graph; $U \subseteq V$: $\overline{\preccurlyeq}$-upw.-cl. set of states.
Result : `true` when U is reachable in $\mathcal{G}$; `false` otherwise.
begin
  Let $\mathcal{T} = \langle N, n_0, B, \Lambda \rangle$ be the tree computed as follows:
  $to\_treat = \{n_0\}$ such that $\Lambda(n_0) = v_i$, $N = \{n_0\}$, $B = \emptyset$;
  while $to\_treat \neq \emptyset$ **do**
    while $to\_treat \neq \emptyset$ **do**
      choose and **remove** n in to_treat ;
      foreach *successor* v *of* $\Lambda(n)$ **do**
        Add n' with $\Lambda(n') = v$ as successor of n;
1   **if** $\nexists n'' \in N : B^*(n'', n') \wedge \Lambda(n') \overline{\preccurlyeq} \Lambda(n'')$ **then** add n' to to_treat;
        else $\mathsf{removed}(n') = \mathtt{true}$;
      Apply reduction rules (see Algorithm 2);
    /* reuse of nodes formerly computed */
2   **while** $\exists n, n' \in N : \neg\mathsf{removed}(n) \wedge \mathsf{removed}(n') \wedge \neg\mathsf{covered}(\Lambda(n'), N) \wedge B(n, n')$ **do** $\mathsf{removed}(n') = \mathtt{false}$;
    /*construction of new nodes */
3   **while** $\exists n \in (N \setminus to\_treat) : \exists v \in V : \Lambda(n) \Rightarrow v \wedge \neg\mathsf{removed}(n) \wedge \neg\mathsf{covered}(v, N)$ **do** add n to to_treat;
  if $\exists n \in N : \mathsf{removed}(n) = \mathtt{false}, \Lambda(n) \in U$ **then return** `true`;
  else return `false`;
end

Algorithm 1: Coverability

while $\exists n, n' \in N : \Lambda(n) \overline{\preccurlyeq} \Lambda(n')$ **do**

1 **if** $\neg B^*(n, n') \wedge \neg$removed$(n) \wedge \neg$removed$(n')$ **then**
  foreach $n'' \in$ nodes(subtree(n)) **do** removed$(n'') =$ `true`;
  $to\_treat \leftarrow to\_treat \setminus$ nodes(subtree(n));

2 **else if** $B^*(n, n') \wedge \neg$removed$(n) \wedge \neg$removed(n') **then**
  Let $S = \{n'' \in$ leaves(subtree$(n')) \mid n'' \notin to_treat\}$;
  Replace_subtree$(n, n', \mathcal{T}, to_treat)$;
  foreach $n'' \in S$ **do**
    if $\nexists n' \in N : B^*(n', n'') \wedge \Lambda(n'') \overline{\preccurlyeq} \Lambda(n')$ **then** add n'' to to_treat;

Algorithm 2: Reduction rules

occurs when nodes are deleted by mistake). The function covered(v, N) returns `true` iff there is a node $n \in N$ with $v \overline{\preccurlyeq} \Lambda(n)$ and removed$(n) =$ `false`.

Sketch of the Algorithm. The tree built by Algorithm 1 is the reachability tree of the $\overline{\preccurlyeq}$-monotonic graph $\mathcal{G}$ on which some reduction rules are applied in order to keep maximal elements only, (so that the labels of the tree eventually computed form a coverability set of $\mathcal{G}$). The sketch of the algorithm is as follows. The inner **while** loop constructs the tree by picking up a node n from to_treat and adding its successors. When a successor n' is smaller than or equal to one of its ancestors n'' ($\Lambda(n') \overline{\preccurlyeq} \Lambda(n'')$), we stop the development of n' (line 1). Then, reduction rules (Algorithm 2) are applied: *(i)* when the tree contains two nodes n and n' such that $\Lambda(n) \overline{\preccurlyeq} \Lambda(n')$ and n is not an ancestor of n', subtree(n) does not need to be developed anymore and is *removed* (that is, all the nodes n'' of subtree(n) are tagged: removed$(n'') =$ `true`, and removed from to_treat); *(ii)* when the tree contains two nodes n and n' such that $\Lambda(n) \overline{\preccurlyeq} \Lambda(n')$ and n is an ancestor of n', we replace subtree(n) by subtree(n'). As mentioned above, the inner **while** loop may fail to compute a coverability set of $\mathcal{G}$ and may only compute an under-approximation. To cope with this problem, we test, at the end of the inner **while** loop whether a coverability set has been computed. More precisely (line 2), we look at all the nodes n' such that removed$(n') =$ `true` and that are direct successors of a node n actually in the tree (i.e.: removed$(n) =$ `false`). When we find that such an n' is not covered by a node actually in the tree, we set removed(n') back to `false`. This step is iterated up to stabilization. Then (line 3), we add into to_treat the nodes n with removed$(n) =$ `false` such that (i) the successor nodes of n have not been developed yet and (ii) there exists one successor v of $\Lambda(n)$ that is not covered by nodes actually in the tree. If to_treat is not empty at the end of these steps, it means that the inner **while** loop has computed an under-approximation of the coverability set. In that case, it is iterated again. Otherwise, when to_treat is empty, it is easy to see that for each node n in the tree such that removed$(n) =$ `false` all the successors of $\Lambda(n)$ are covered by nodes n' of the tree such that removed$(n') =$ `false`. Since the root node of the tree covers v_i, we conclude that $\{v \mid \exists$ a node n of the tree: $\Lambda(n) = v,$ removed$(n) =$ `false`$\}$ is a coverability set.

The next theorem states the correctness of Algorithm 1:

Theorem 3. *Algorithm 1, when applied to the $\overline{\preccurlyeq}$-monotonic graph $\mathcal{G}$ and the $\overline{\preccurlyeq}$-upward closed set U, always terminates and returns* `true` *if and only if there exists a node $v \in U$ such that v is reachable from v_i in $\mathcal{G}$.*

Remark 2. When $\Rightarrow$ is computable, Algorithm 1 can compute $\mathcal{T}$ without disposing of the whole graph $\mathcal{G}$ (v_i only is necessary). In that case, Algorithm 1 efficiently explores a (possibly small) portion of a (potentially large) $\overline{\preccurlyeq}$-monotonic graph without building it entirely.

Application to EEC. Thanks to Proposition 5, we can apply Algorithm 1 to any EPRG built at the 'Expand' phase, if the WSTS is lossy (but by Proposition 3, we can always take the lossy version of any WSTS), and the sets C_i are $\preccurlyeq_e$-downward-closed. We show in Section 5 that this is not restrictive in practice.

Algorithm 1 is also useful to improve the 'Enlarge' phase in the case where the And-Or graph is *degenerated*. An And-Or graph is degenerated whenever each Or-node has only one successor. Hence a degenerated And-Or graph $G = \langle V_A, V_O, v_i, \Rightarrow\rangle$ is equivalent to a plain graph $G' = \langle V_O, v_i, \Rightarrow'\rangle$ where we have $v \Rightarrow' v'$ if an only if $\exists v'' \in V_A : v \Rightarrow v'' \Rightarrow v'$. From Proposition 4, G' is a $\sqsubseteq$-monotonic graph, for any WSTS with adequate domain of limits $\langle L, \sqsubseteq, \gamma\rangle$.

5 Expand, Enlarge and Check in Practice

Checking the practical usability of EEC by implementing it is an essential step. Indeed, even if we dispose of a nice theoretical result that shows the completeness of EEC, the theoretical complexity of the problems addressed here remain *non-primitive recursive* [13]. In this section, we specialize EEC to obtain efficient procedures for the coverability problem on two classes of WSTS: the monotonic extensions of Petri nets (EPN) and the lossy channel systems (LCS).

Since And-Or graphs for EPN are always degenerated [11], we can exploit the efficient procedure described in Section 4 in both the 'Expand' and the 'Enlarge' phase. As far as LCS are concerned, the main difficulty relies in the construction of the And-Or graph: the 'Expand' phase requests an efficient procedure to compute the most precise successors of any Or-node.

5.1 Extended Petri Nets

We consider monotonic extensions of the well-known Petri net model (such as Petri nets with transfer arcs, a.s.o., see [4]). Due to the lack of space, we refer the reader to [11] for the syntax. An EPN P defines a WSTS $S = \langle \mathbb{N}^k, \mathbf{m}_0, \rightarrow\rangle$ where k is the number of places of P, $\mathbf{m}_0$ is the initial marking of P and $\rightarrow \subseteq \mathbb{N}^k \times \mathbb{N}^k$ is a transition relation induced by the transitions of the EPN (see [11] for details).

Domain of Limits. To apply the schema of algorithm to extensions of Petri nets, we proposed in [11] to consider the domain of limits $\langle \mathcal{L}, \preccurlyeq_e, \gamma\rangle$ where $\mathcal{L} = (\mathbb{N} \cup \{+\infty\})^k \setminus \mathbb{N}^k$, $\preccurlyeq_e \subseteq (\mathbb{N} \cup \{+\infty\})^k \times (\mathbb{N} \cup \{+\infty\})^k$ is such that $\langle m_1, \ldots, m_k\rangle \preccurlyeq_e$

$\langle m'_1,\ldots,m'_k\rangle$ if and only if $\forall 1 \leq i \leq k : m_i \leq m'_i$ (where $\leq$ is the natural order over $\mathbb{N}\cup\{+\infty\}$. In particular: $c < +\infty$ for all $c \in \mathbb{N}$). γ is defined as: $\gamma(\mathbf{m}) = \{\mathbf{m}' \in \mathbb{N}^k \mid \mathbf{m}' \preccurlyeq_e \mathbf{m}\}$. The sequences of C_i's and L_i's are defined as follows: (D_1) $C_i = \{0,\ldots,i\}^k \cup \{\mathbf{m} | \mathbf{m} \preccurlyeq_e \mathbf{m}_0\}$, i.e. C_i is the set of markings where each place is bounded by i (plus the $\preccurlyeq_e$-downward closure of the initial marking); (D_2) $L_i = \{\mathbf{m} \in \{0,\ldots i,+\infty\}^k \mid \mathbf{m} \notin \mathbb{N}^k\}$.

Efficient Algorithm. To achieve an efficient implementation of EEC, and according to Proposition 3, we consider the lossy version of EPN (that are lossy WSTS) to decide the coverability problem on EPN. As the sets C_i are $\preccurlyeq_e$-downward-closed, we use the algorithm of Section 4 to efficiently compute the 'Expand' phase.

The 'Enlarge' phase is improved by using the method of [11] to compute the successors of any Or-node of the And-Or graph. Moreover, the And-Or graphs are always degenerated in the case of (lossy) EPN [11], hence we also apply Algorithm 1 during that phase. Note that, although the set of successors of a state of a lossy EPN can be large, $\preccurlyeq_e$-monotonic graphs and And-Or graphs allow us to consider the maximal successors only.

Experiments. We have implemented the techniques described so far in a prototype capable of analyzing EPN. We have run the prototype on about 30 examples[1] from the literature. Table 1 reports on selected results. The case studies retained here are mainly abstractions of multi-threaded Java programs (from [14]).

When applied to these examples, the basic symbolic backward algorithm of [1] seldom produces a result within the time limit of 1 hour of CPU time we have fixed (column **Pre**). A heuristic presented in [5] uses place-invariants to guide the search and improves the performance of the prototype (which has been fined tuned during several years of research). Still, it might not terminate on some examples (column **Pre+Inv**). On the other hand, we have implemented EEC with a basic exploration of the abstractions (column $\mathbf{EEC}_1$). The performance increase is already noticeable when compared to the basic backward approach. Moreover, when we apply the efficient exploration presented in Section 4, our prototype perfoms much better, on all the examples (column $\mathbf{EEC}_2$). Our experiments prove the practical superiority of the forward analysis at work in EEC.

Other tools such as FAST and LASH can analyze the same examples by using a forward procedure. Remark that these tools can handle a broader class of systems than EEC. In practice, FAST does not always terminate on our examples[2].

5.2 Lossy Channel Systems

Lossy channel systems (LCS) are systems made up of a finite number of automata which communicate through lossy FIFO channels, by writing to or reading from the channels when a transition is fired. This model is well-studied, see e.g. [3, 2]. In particular, the Simple Regular Expressions (sre), a symbolic representation

[1] See `http://www.ulb.ac.be/di/ssd/ggeeraer/eec`

[2] See `http://www.lsv.ens-cachan.fr/fast/example.php`.

Table 1. results obtained on Intel Xeon 3Ghz with 4Gb of memory : **cat.**: category of example (PNT = Petri nets with transfer arcs, PN = (unbounded) Petri net); **P**: number of places; **T**: number of transitions; **EEC**$_1$: basic EEC with complete exploration of the graph; **EEC**$_2$: EEC with efficient exploration of Section 4; **Pre + Inv**: Backward approach with invariants; **Pre**: same without invariants. Times in second

Example					EEC$_1$	EEC$_2$		Pre+Inv	Pre
cat.	**name**	**P**	**T**	**Safe**	**Time**	**Time**	**Mem (Kb)**	**Time**	**Time**
PNT	Java	44	37	×	10.39	8.47	23,852	1.40	1276.56
PNT	delegatebuffer	50	52	√	↑↑	180.78	116,608	↑↑	↑↑
PNT	queuedbusyflag	80	104	√	341	28.87	21,388	↑↑	↑↑
PN	pncsacover	31	36	×	800	7.54	13,704	40.83	↑↑

for downward-closed sets of states of LCS, have been defined. Algorithms to symbolically compute classical operations, such as the union, intersection or the Post, have been devised. In the sequel, we will rely on this background.

Preliminaries. In order to keep the following discussion compact, we will consider, without loss of generality, a LCS $\mathcal{C}$ made up of a single automaton (with set of states Q) and a single FIFO channel (initially empty, with alphabet Σ). A state of $\mathcal{C}$ is a pair $\langle q, w\rangle$, where $q \in Q$ is the state of the automaton, and $w \in \Sigma^*$ is the content of the channel. Let $S_{\mathcal{C}}$ be the set of states of $\mathcal{C}$. A transition of $\mathcal{C}$ is of the form $\langle s_1, Op, s_2\rangle$ where $s_1, s_2 \in Q$ and Op is $!a$ (add a to the channel), or $?a$ (consume a on the channel), or *nop* (no modification of the channel), for any $a \in \Sigma$. The semantics is the classical one, see [3]. The w.q.o. $\preccurlyeq_w \subseteq \Sigma^* \times \Sigma^*$ is defined as follows: $w_1 \preccurlyeq_w w_2$ iff w_1 is a (non-necessarily contiguous) subword of w_2. A *downward-closed regular expression* (dc-re) is a regular expression that is either $(a+\varepsilon)$ for some $a \in \Sigma$, or $(a_1+a_2+\ldots+a_n)^*$ for $\{a_1, a_2, \ldots, a_n\} \subseteq \Sigma$. Given a dc-re d, $\alpha(d)$ (the *alphabet of* d) is the set of all the elements of Σ that occur in d. A *product* (of dc-re) is either ε or an expression of the form $d_1 \cdot d_2 \cdot \ldots \cdot d_n$, where d_1, d_2,..., d_n are dc-re. Given a product p, $[\![p]\!] \subseteq \Sigma^*$ denotes the ($\preccurlyeq_w$-downward-closed) language generated by p, and $|p|$, denotes its size, i.e., the number of dc-re that compose it (for $w \in \Sigma^*$, $|w|$ is defined the usual way). Let $P(\Sigma)$ denote the set of all products built from Σ.

Domain of Limits. Let $\mathcal{L}(\Sigma, Q)$ denote the set of limits $\{\langle q, p\rangle | q \in Q, p \in P(\Sigma)\} \cup \{\top\}$. For any $\langle q, p\rangle \in \mathcal{L}(\Sigma, Q)$, $[\![\langle q, p\rangle]\!]$ denotes the set of states $\langle q, w\rangle \in \mathcal{S}_{\mathcal{C}}$ such that $w \in [\![p]\!]$. We define the function $\gamma : \mathcal{L}(\Sigma, Q) \to 2^{\mathcal{S}_{\mathcal{C}}}$ such that (*i*) $\gamma(\top) = Q \times \Sigma^*$ and (*ii*) $\gamma(\langle q, p\rangle) = [\![\langle q, p\rangle]\!]$, for any $\langle q, p\rangle \in \mathcal{L}(\Sigma, Q) \setminus \{\top\}$.We define $\overline{\sqsubseteq} \subseteq \mathcal{L}(\Sigma, Q) \times \mathcal{L}(\Sigma, Q)$ as follows : $c_1 \overline{\sqsubseteq} c_2$ if and only if $\gamma(c_1) \subseteq \gamma(c_2)$. When $c_1 \overline{\sqsubseteq} c_2$ but $c_2 \overline{\not\sqsubseteq} c_1$, we write $c_1 \overline{\sqsubset} c_2$.

Let us now define the sets of concrete and limit elements we will consider at each step. We define $C_i = \{\langle q, w\rangle \mid \langle q, w\rangle \in \mathcal{S}_{\mathcal{C}}, |w| \leq i\}$, i.e. C_i is the set of states where the channel contains at most i characters. Similarly, we define L_i as follows: $L_i = \{\langle q, p\rangle \in \mathcal{L}(\Sigma, Q) \mid |p| \leq i\} \cup \{\top\}$, i.e. L_i contains the limits where a product of length at most i represents the channel (plus $\top$).

Efficient Algorithm. In the case of LCS, the And-Or graph one obtains is, in general, not degenerated. Hence, the techniques presented in Section 4 can be used along the 'Expand' phase only (the C_is are $\preccurlyeq_w$-downward-closed and the WSTS induced by LCS are lossy). In the sequel, we try nonetheless to improve the 'Enlarge' phase by showing how to directly compute (i.e. without enumeration of states) the set of (most precise) successors of any Or-node. Notice that following the semantics of LCS, the Post operation can add at most one character to the channel. Hence, we only need to be able to approximate precisely any $c \in L_{i+1} \cup C_{i+1}$ by elements in $L_i \cup C_i$.

Over-Approximation of a Product. Given a product $p \neq \varepsilon$ and a natural number $i \geq 1$ such that $|p| \leq i+1$, let us show how to directly compute, the most complete and most precise set of products that over-approximate p, and whose size is at most i. For this purpose, we first define an auxiliary function $L(p)$. Let $p = d_1 \cdot d_2 \cdots d_n$ be a product.

$L(p) = \bigcup_{1 \leq i \leq n-1} \{d_1 \cdots d_{i-1} \cdot (c_1 + \ldots + c_m)^* \cdot d_{i+2} \cdots d_n | \{c_1, \ldots, c_m\} = \alpha(d_i) \cup \alpha(d_{i+1})\}$. We can now define $\mathsf{Approx}(p,i)$ for $|p| \leq i+1$ and $i \geq 1$. $\mathsf{Approx}(p,i) = \{p\}$ when $|p| \leq i$, and $\mathsf{Approx}(p,i) = \{q \in L(p) \mid \nexists q' \in L(p) : q' \overline{\sqsubseteq} q\}$ when $|p| = i+1$.

Proposition 6. *Given a natural number i and a product of* dc-re *p such that $|p| \leq i+1$, for all products of* dc-re *p' such that (i) $[\![p]\!] \subseteq [\![p']\!]$; (ii) $|p'| \leq i$ and (iii) $p' \notin \mathsf{Approx}(p,i)$: $\exists p'' \in \mathsf{Approx}(p,i) : [\![p'']\!] \subseteq [\![p']\!]$.*

Hence, Approx allows us to over-approximate any limit element of L_{i+1} by elements of L_i. In order to handle elements of C_{i+1}, we extend the definition of Approx as follows. Let i be a natural number and $w = a_1 \ldots a_n \in \Sigma^*$ (with $n \leq i+1$) be a word, then $\mathsf{Approx}(w,i) = w$ when $n \leq i$, and $\mathsf{Approx}(w,i) = \mathsf{Approx}(p_w, i)$ with $p_w = (a_1 + \varepsilon) \cdots (a_n + \varepsilon)$ otherwise. Remark that w and p_w both define the same $\preccurlyeq_w$-downward-closed set, and Proposition 6 remains valid.

When the LCS has more than one channel, a state (limit) associates a word (or a product of dc-re) to each channel. It that case, the best approximation can be computed by taking the product of the best approximations for each channel.

Experiments. We have built a prototype to decide the coverability problem for LCS. It implements the improvements of the 'Expand' and 'Enlarge' phases presented above. Another improvement in the construction of the And-Or graph consists in computing only the states that are reachable from the initial state.

Table 2. Results obtained on Intel Xeon 3Ghz with 4Gb of memory : **S** and **E**: number of states and edges of the graph ; **C**: number of channels; **EEC**: execution time (in second) of an implantation of EEC

Case study	S	E	C	EEC	Case study	S	E	C	EEC
ABP	48	192	2	0.18	BRP_3	480	2,460	2	0.35
BRP_1	480	2,460	2	0.19	BRP_4	480	2,460	2	0.41
BRP_2	480	2,460	2	0.19	BRP_5	640	3,370	2	0.19

Table 2 reports on the performance of the prototype when applied to various examples of the literature: the Alternating Bit Protocol (ABP), and the Bounded Retransmission Protocol (BRP), on which we verify five different properties [3]. Table 2 shows very promising results with our simple prototype.

6 Conclusion

In this paper we have pursued a line of research initiated in [11] with the introduction of the 'Expand, Enlarge and Check' algorithm. We have shown in the present work that, for a peculiar subclass of WSTS, one can derive efficient practical algorithms from this theoretical framework. We have presented an efficient method to decide the coverability problem on monotonic graphs. This solution fixes a bug, for the finite case, in the minimal coverability tree algorithm of [8]. It can always be applied to improve the 'Expand' phase. In the case of *extended Petri nets*, it can also be used to improve the 'Enlarge' phase. In the case of *lossy channel systems*, we have also shown how to improve the 'Expand' phase, by building the And-Or graph in an efficient way. We have implemented these techniques in two prototypes, working in a *forward* fashion. Their excellent behaviours clearly demonstrate the practical interest of EEC.

Acknowledgements. We are deeply grateful to Ahmed Bouajjani and Mihaela Sighireanu, who have given us access to their C++ library to manipulate sre.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General Decidability Theorems for Infinite-state Systems. In *Proc. of LICS'96*, pages 313–321. IEEE, 1996.
2. Parosh Abdulla, Aurore Annichini, and Ahmed Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proc. of TACAS'99*, number 1579 in LNCS, pages 208–222. Springer-Verlag, 1999.
3. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Form. Methods Syst. Des.*, 25(1):39–65, 2004.
4. G. Ciardo. Petri nets with marking-dependent arc multiplicity: properties and analysis. In *Proc. of ICATPN'94*, vol. 815 of *LNCS*, pages 179–198. Springer, 1994.
5. G. Delzanno, J.-F. Raskin, and L. Van Begin. Attacking Symbolic State Explosion. In *Proc. of CAV 2001*, vol. 2102 of *LNCS*, pages 298–310. Springer, 2001.
6. E. A. Emerson and K. S. Namjoshi. On Model Checking for Non-deterministic Infinite-state Systems. In *Proc. of LICS '98*, pages 70–80. IEEE, 1998.
7. J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *Proc. of LICS'99*, pages 352–359. IEEE, 1999.
8. A. Finkel. The minimal coverability graph for Petri nets. In *Proc. of APN'93*, vol. 674 of *LNCS*, pages 210–243. Springer, 1993.
9. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

10. A. Finkel G. Geeraerts, J.-F. Raskin, and L. Van Begin. A counter-example to the minimal coverability tree algorithm *Technical report ULB 535*. `http://www.ulb.ac.be/di/ssd/ggeeraer/papers/FGRV05-Coverability.pdf`
11. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS. In *Proc. of FSTTCS'04*, vol. 3328 of LNCS, pages 287–298. Springer-Verlag, 2004.
12. T. A. Henzinger, O. Kupferman, and S. Qadeer. From *pre*historic to *post*modern symbolic model checking. *Formal Methods in System Design*, 23(3):303–327, 2003.
13. Ph. Schnoebelen. Verifying Lossy Channel Systems has Nonprimitive Recursive Complexity. *Information Processing Letters*, 83(5), 251–261, 2002
14. L. Van Begin. Efficient Verification of Counting Abstractions for Parametric systems. *PhD thesis*, Université Libre de Bruxelles, Belgium, 2003.