Noname manuscript No.
(will be inserted by the editor)

A Unified View of Parameterized Verification of Abstract Models of Broadcast Communication

Giorgio Delzanno

the date of receipt and acceptance should be inserted later

Abstract We give a unified view of different parameterized models of concurrent and distributed systems with broadcast communication based on transition systems. Based on the resulting formal models, we discuss related verification methods and tools based on abstractions and symbolic state exploration.

1 Introduction

Specifications of concurrent and distributed system are often formulated for a finite but arbitrary number of processes. Typical examples are consistency protocols, e.g., cache coherence protocols used in hardware or in distributed systems, solutions to the critical section problem, and distributed algorithms. In this context Parameterized Verification (PV) is a research field aimed at designing automated or semi-automated verification methods for systems composed of families of replicated components. Formal specification languages for concurrent systems, e.g. Petri nets, provide constructs to represent configurations in which either the number of components is not fixed a priori or it can change dynamically. Lifting verification problems from finite-state to parameterized systems often leads to undecidable decision problems or to problems with high complexity. For instance, it has been shown that model checking for parameterized system is undecidable [21]. Undecidability holds even for apparently simple classes of systems like Boolean programs with shared variables [94]. The above mentioned general undecidability results make the problem challenging for automated reasoning techniques. In this context restricted classes of systems and properties have been considered in order to identify complete methods or effective procedure with low complexity.

In this paper we will focus on a class of formal models for concurrent and distributed systems in which synchronization is achieved by using broadcast communication, a less standard communication primitive than rendez-vous or point-to-point communication. Rendez-vous communication involves a fixed a priori number of agents (e.g. a sender and a receiver in point-to-point communication). Properties of rendez-vous communication have been investigated deeply in the field of automated verification, see e.g. [72]. Interactions in models with broadcast communication are usually defined by allowing a finite but arbitrary number of agents to react to a given message or signal. This type of communication is particularly useful to define protocols for replicated systems, e.g. cache coherence protocols, algorithms with global conditions, e.g., simultaneous resets of local variables, and communication in an open environment like an Ad Hoc or Wireless network. Algorithmic verification of models with broadcast communication started receiving more attention after the introduction of the Broadcast Protocols of Emerson and Namjoshi [57]. Several interesting properties have been obtained in this setting by transferring results coming from the theory of Petri nets and of Well-structured Transition Systems [66,16]. Furthermore, the interpretation of broadcast communication in terms of whole place operations in Petri nets have inspired several extensions of the original model obtained, e.g., by adding time, data, communication links, message buffers, and communication groups. In the paper we will give a uniform presentation of models of increasing complexity by using transition systems to formally specify their semantics. For each of the proposed model we will study the impact of the above mentioned features on decid-

DIBRIS, Università di Genova, Italy
E-mail: Giorgio.Delzanno@unige.it

ability and complexity of verification problems formulated for arbitrary number of processes.

Plan of the paper

In Section 2 we present preliminary notions (e.g. Petri nets and Vector Addition Systems with State (VASS), Well-structured Transition Systems (WSTS)) that are used for describing computability and complexity of verification problems of the different models described in the rest of the paper. In Section 3 we introduce a basic model for broadcast communication called Broadcast Protocols that extends Petri nets with whole place operations. In Section 4 we extend Broadcast Protocols in order to model processes that carry data. In Section 5 we introduce models of distributed systems in which the network is modeled in an explicit way using graphs with a combination of different features like clocks and data. For all above mentioned models, we present decidability and undecidability results for verification of safety and, in some cases, liveness properties. In Section 6 we describe some existing tools that can be used for parameterized verification. Finally, in Section 7 we address conclusions and future work.

2 Basic Models and Parameterized Verification Methods

Let A be a finite set. We use $A^{\oplus}$ to denote the class of multisets with elements in A. A multiset m can be viewed as a map from A to $\mathbb{N}$ s.t. $m(a) \geqslant 0$ denotes the number of occurrences of a in m. We use here $\subseteq$ to denote multiset inclusion, i.e., $m_1 \subseteq m_2$ if $m_1(a) \leqslant m_2(a)$ for any $a \in A$, $\oplus$ to denote multiset union, i.e., $(m_1 \oplus m_2)(a) = m_1(a) + m_2(a)$ for any $a \in A$, and $\ominus$ to denote multiset difference, i.e., for any $a \in A$ $(m_1 \ominus m_2)(a) = m_1(a) - m_2(a)$ if $m_1(a) \geqslant m_2(a)$, $= 0$ otherwise.

A quasi ordering $\langle S, \leqslant \rangle$ is a reflexive and transitive ordering. Given a quasi ordering $\langle S, \leqslant \rangle$, an upward closed set of states is a subset $U \subseteq S$ such that for any $s \in U$, if $s \leqslant s'$ then $s' \in U$. Given a set $B \subseteq S$, we say that B generates the upward closed set $B\uparrow$ defined as $\{s \mid s' \in B,\ s' \leqslant s\}$. A well quasi ordering (wqo) $\langle S, \leqslant \rangle$ is a quasi ordering such that for every infinite sequence of elements $s_1 s_2 \ldots$ there exist $i < j$ such that $s_i \leqslant s_j$. A wqo has the finite basis property, i.e., every upward closed set $U \subseteq S$ is generated by a finite set B. Examples of wqo's are listed below.

- Finite sets equipped with equality.
- Natural numbers equipped with the less than or equal ordering.
- Tuples of natural numbers equipped with the point-wise $\leqslant$ ordering, i.e., $\langle a_1, \ldots, a_n \rangle \leqslant \langle b_1, \ldots, b_n \rangle$ if $a_i \leqslant b_i$ for $i : 1, \ldots, n$ (Dickson's Lemma [53]).
- Tuples of natural numbers extended with ω equipped with the point-wise $\leqslant$ ordering. Here we assume that $a \leqslant \omega$ for any $a \in N$, i.e., ω represents the set of natural numbers.
- Multisets over a finite alphabet equipped with multiset inclusion.
- Words over a finite alphabet equipped with the subword ordering.
- Sets, multisets, and tuples of elements taken from a wqo $\langle S, \leqslant \rangle$ equipped with an embedding (Higman's Lemma [75]). An embedding is an injection h from m_1 to m_2 s.t. for any $a \in m_1$ if $h(a) = b \in m_2$ then $a \leqslant b$.
- Words with elements in a wqo $\langle S, \leqslant \rangle$ equipped with an injective and monotone embedding.

Dickson's lemma is a central result for defining decision procedures of Petri nets. The above mentioned properties of wqo's can be used to build new wqo's. For instance, Higman's lemma can be used to prove that words of multisets of elements taken from a finite set equipped with a monotone and injecting embedding form a wqo. This kind of ordering has been applied to obtain decidability results for timed extensions of Petri nets [15].

Consider now the ordering $\langle \mathcal{P}_f(S), \leqslant \rangle$ built over finite sets with elements in S as follows. For sets $A, B \in \mathcal{P}_f(S)$, $A \sqsubseteq B$ if for every $b \in B$ there exists $a \in A$ s.t. $a \leqslant b$. As shown by the counterexample based on Rado's structure [2], the ordering $\sqsubseteq$ is not always a wqo. However, in [16] it has been proved that for an increasing chain $A_1 \subseteq A_2 \subseteq A_3 \ldots$ of sets in $\mathcal{P}_f(S)$, there exist i, j s.t. $A_i \sqsubseteq A_j$. This property is particularly useful to prove termination of fixpoint algorithms in which intermediate results are maintained in memory and compared via a wqo (for eliminating redundancies and detect fixpoints).

Let S be a possibly infinite set of configurations. A transition system is a tuple $\langle S, \rightarrow, s_0 \rangle$ such that $\rightarrow \subseteq S \times S$ is the transition relation, and s_0 is the initial state. We use $s_1 \rightarrow s_2$ to denote a pair $\langle s_1, s_2 \rangle \in \rightarrow$. A computation is a sequence of states $s_0 s_1 s_2 \ldots$ s.t. $s_i \rightarrow s_{i+1}$ for $i \geqslant 0$. The set of successors of a set of configurations A is defined as $post(A) = \{t \mid s \rightarrow t,\ s \in A\}$. The whole set of successor states of a set of configurations A is defined as $post^*(A) = \bigcup_{i \geqslant 0} post^i(A)$, where $post^0(A) = A$, and $post^{i+1}(A) = post(post^i(A))$ for $i \geqslant 0$. The Reachability Set of a transition system is usually defined as $post^*(I)$, where I contains the initial state s_0, or, more in a general a set of initial states. The nodes of the reachability graph are labeled with config-

urations of the reachability set. Edges correspond to the relation $\rightarrow$.

The set of predecessor states of a set of configuration A is defined as $pre(A) = \{s \mid s \rightarrow t,\ t \in A\}$. The whole set of predecessor states of a set of configurations A is defined as $pre^*(A) = \bigcup_{i \geqslant 0} pre^i(A)$, where $pre^0(A) = A$, and $pre^{i+1}(A) = pre(pre^i(A))$ for $i \geqslant 0$.

Transition systems are used to define the operational semantics of models of computations like programs, concurrent and reactive systems, etc. In this paper we will use the notation $\rightarrow$ to denote the transition relations of different types of models and $\rightarrow^*$ to denote the reflexive and transitive closure of $\rightarrow$.

A transition system is said to be monotone with respect to an ordering $\langle S, \leqslant \rangle$ on configurations if for any state s_1, s_2, s_3 if $s_1 \rightarrow s_2$ and $s_1 \leqslant s_3$, then there exists s_4 s.t.$s_2 \leqslant s_4$ and $s_3 \rightarrow s_4$. Monotonicity is an important property to obtain decidability results for infinite-state transition systems.

Let $\langle S, \rightarrow, s_0 \rangle$ be a transition system. Given s_0 and s_1, the *Reachability Problem* consists in deciding whether $s_0 \rightarrow^* s_1$.

Given states s_0 and s_1, the *Coverability Problem* consists in deciding whether there exists a state s_2 s.t. $s_1 \leqslant s_2$ and $s_0 \rightarrow^* s_2$. Coverability is strictly related to the verification of safety properties. If bad states can be represented via an upward closed set generated by state s_1 then a decision procedure for Coverability can be used to detect possible error traces or to prove absence of errors.

More general verification problems on transition systems can be formulated using temporal logic like LTL (linear temporal logic). In this setting formulas are built on top of classical connectives enriched with temporal operators like X (next), F (eventually) and G (always) interpreted over paths of the reachability graph of a transition system.

2.1 Vector Addition Systems (VAS)

Vector addition systems (VAS) are infinite-state transition systems in which configurations are tuples of fixed size of natural numbers. We use $\mathbf{u} = \langle u_1, \ldots, u_n \rangle$ to denote a tuple and $\mathbf{u} \leqslant \mathbf{v}$ to denote the point-wise ordering of tuples. A transition is defined by a guard and an action. Let $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ be a tuple of variables that denote the current value of the counters, and let $\mathbf{x}' = \langle x'_1, \ldots, x'_n \rangle$ be a tuple of variables that denote the value of the counters in the next state. Guards are conjunctions of comparisons of the form $\mathbf{x_i} \geqslant c_i$ where $x_i \in \mathbf{x}$ and c_i is a natural number. Actions are defined by using affine transformations of the form $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$, where $\delta = \langle d_1, \ldots, d_n \rangle$ is a n-vector of integer constants. A transition with guard $\mathbf{x} \geqslant \mathbf{c}$, and action $\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}$ is enabled in the state $\mathbf{u}$ if $\mathbf{c} \leqslant \mathbf{u}$. The application of the transition is then $\mathbf{u} \rightarrow \mathbf{u} + \boldsymbol{\delta}$.

The Reachability Problem for VAS is known to be decidable in EXPSPACE [87,90]. The original proof is not applicable in practice. A recent reformulation of Leroux [85,86] yields new insights into Presburger definable over- and under-approximations of the problem. The Coverability Problem can be decided via forward exploration with accelerations, an algorithm due to Karp and Miller [80]. Before illustrating how the Karp-Miller construction works, let us first observe that the reachability set of a VAS may be infinite. A finite over-approximation of the reachability set can be obtained by applying accelerations during the construction of the reachability graph. The acceleration is applied along every path of the reachability graph. Consider a state $\mathbf{u} = \langle u_1, \ldots, u_n \rangle$ with predecessor states $\mathbf{v^i} = \langle v_1^i, \ldots, v_n^i \rangle$ along a given path. For some i, if $\mathbf{v^i} \leqslant \mathbf{u}$, then we replace with ω all components u_j of $\mathbf{u}$ s.t. $v_j^i < u_j$. Coverability can be checker by comparing the target configurations $\mathbf{t}$ with the extended states in the resulting graph. If one of the extended states is larger than $\mathbf{t}$, coverability holds for $\mathbf{t}$. Termination of the resulting algorithm is ensured by the wqo property of tuples of natural numbers extended with ω that ensures that there cannot be paths containing an infinite sequence of incomparable extended states. The coverability graph can be applied to verify liveness properties by first searching for strongly connected components and then for vectors that are covered by the vector that must be visited infinitely often (Repeated Coverability).

VASS are systems equipped with a control state ranging from a finite set and a vector of counters ranging over natural numbers. VASS can be reduced to VAS by encoding control states as special values of an additional counter.

2.2 VAS for Counting Abstractions

In [72] German and Sistla provide an interesting example of applications of VAS to reason on families of concurrent systems. In this setting individual processes are finite-state automata with special synchronization labels, namely ℓ and $\overline{\ell}$. A global state is a tuple $\langle q_1, \ldots, q_m \rangle$ of states of individual processes such that q_i is taken from a finite set of states Q for $i : 1, \ldots, m$. Local transitions have the effect of updating the state of an individual process. In a local step, a successor $\gamma' = \langle q'_1, \ldots, q'_n \rangle$ of a configuration $\gamma = \langle q_1, \ldots, q_n \rangle$ is obtained by selecting a process i, applying a local transition $q_i \rightarrow q'_i$, and,

finally, by requiring that $q'_j = q_j$ for $j \neq i$. Rendez-vous transitions simultaneously update the local states of a pair of processes. Specifically, a rendez-vous step is obtained by selecting two process i, j, by applying the transitions $q_i \rightarrow q'_i$ and $q_j \rightarrow q'_j$, and by requiring that $q'_k = q_k$ for $k \neq i, j$.

In [72] German and Sistla applied the so called counting abstraction in order to reduce parameterized verification problems for LTL formulas to verification problems for VASS. The counting abstraction consists in abstracting global states using a finite set of counters, one for each state of an individual process. Counters are used to keep track of state updates of individual processes. They keep track of the number of processes in each state at every step of a computation. A control state is used to model controller processes. By using the general results on VASS, it is possible to reduce LTL verification for propositions associated to the controller to (Repeated) Coverability.

2.3 Petri nets

A Petri net is a tuple $\langle P, T, F, M_0 \rangle$ where P is a finite set of places, T is a finite set of transitions and F is a set of arcs, i.e., $F \subseteq (P \times T) \cup (T \times P)$ and M_0 is the initial marking. A marking is a multiset $M : P \rightarrow N$. When $M(p) = k$, we say that place p contains k tokens. A marking represents the current configuration of a Petri net. Transitions describe possible updates of the current configuration of a Petri net. The set of places with outgoing edges pointing to t is denoted ${}^\bullet t$. The set of places with incoming edges from transition t is denoted $t^\bullet$. Transitions can be generalized in order to label arcs with multiplicity, i.e., ${}^\bullet t$ and $t^\bullet$ become multisets on P. We say that t is enabled at marking M if ${}^\bullet t \subseteq M$, i.e., for each place of P, M has at least as many tokens as ${}^\bullet t$. If t is enabled in M, the firing of t yields a new marking M' defined as $M' = (M \ominus {}^\bullet t) \oplus t^\bullet$. Namely, the tokens in ${}^\bullet t$ are removed from M and those in $t^\bullet$ are added to the resulting multiset.

A marking can also be viewed as an abstract representation of a global configuration of a concurrent system in which we only maintain the number of processes in a finite set of possible states (the set of places). Petri net markings can be viewed as vectors of natural numbers. Petri net transitions can then be interpreted as transitions of VAS. Decidability of VAS reachability can naturally be transferred to Petri nets. Recently, it has been proved that the reachability problem for Petri nets remains decidable for Petri nets with one inhibitor arc, i.e., a guard that checks whether a place is empty [95]. A similar result holds for VAS with one zero test [28]. Petri nets with two inhibitor arcs are Turing equivalent. Coverability is still decidable in Petri nets with reset and transfer arcs [55]. Reset arcs remove all tokens from a give place. Transfer arcs move all tokens from one place to another. When modeled in a VAS, reset arcs are equalities of the form $x' = 0$ for a given counter x. Transfer arcs are equalities of the form $x' = x + y, y' = 0$ for counters x, y.

The use of Petri nets for modeling synchronization primitives in high level programming languages has been proposed e.g. in [22]. In this setting a Boolean abstraction is extracted from a concurrent program without recursion but with synchronization primitives like locks and monitors. In this abstraction global and local variables are restricted so as to range over finite domains only. A counting abstraction is then applied in order to use counters to maintain the number of threads in a given program location during the execution of the original program.

2.4 Well-structured Transition Systems

Well structured transitions systems (wsts) [16,66] are transition systems in which it is possible to automatically verify the Coverability problem. More specifically, a transition system is well structured if: (1) it is monotone w.r.t. a wqo $\leqslant$ on configurations, (2) given a basis B of an upward closed set of configurations U, it is possible to algorithmically compute a basis B' of the set of predecessor states $pre(U)$ of U, (3) it is possible to algorithmically check whether s_0 belongs or not to an upward closed set of configurations.

In [16,66] it has been proved that Coverability is decidable for wsts's. The algorithm that decides coverability is based on a backward reachability analysis in which minimal elements of upward closed sets of states are used to symbolically represent infinite sets of states. Let Φ be a set of finite bases of upward closed sets. We use $\Phi\uparrow$ to denote the upward closure of elements in Φ. We define the sequence $\Phi_0\Phi_1 \ldots$ as follows:

- $\Phi_0 = \{s_1\}$
- $\Phi_{i+1} = \Phi_i \cup spre(\Phi_i)$ for $i \geqslant 0$

where, in general, $spre(\Phi)$ computes a set of minimal elements that represent $pre(\Phi_i\uparrow)$.

As shown in [16], since $\leqslant$ is a wqo, then the chain necessarily stabilizes, i.e., there exists k s.t. Φ_{k+1} represents the same set of configurations as Φ_k (i.e. Φ_k is a least fixpoint). When a fixpoint has been detected, it remains to check whether the initial states belong to $\Phi_k\uparrow$.

2.5 Petri nets as Wsts

The theory of wsts provides alternative ways to define decision procedures for VAS (and Petri nets). Instead of using forward exploration with accelerations as in the Karp-Miller construction, we can apply the above mentioned backward algorithm starting from the target vector $\mathbf{v}$ (marking) viewed as a generator for the infinite set of vectors $\{\mathbf{u} \mid \mathbf{v} \leqslant \mathbf{u}\}$. Predecessor can be computed symbolically by applying the transitions backwards on the generator $\mathbf{v}$ and then conjoining the results with the guard of the transition. Symbolic representations of upward closed sets via linear constraints or dedicated data structures have been considered, e.g., in [40,44]. The EPR theory of arrays used in SMT solvers can also be used to symbolically represent upward closed set of configurations of parameterized systems [73].

Heuristics to reduce the number of minimal elements used to represent upward closed sets of vectors have been considered in [78]. The algorithm is based on non-deterministic guesses of new minimal elements (smaller than those present in the current analysis) that are removed in backtracking in case their selection leads to incorrect results. A general forward exploration algorithm for well structured transition systems has been considered in the Expand, Enlarge and Check framework [70]. The algorithm is based on an abstraction refinement schema in which the abstract domain is made more and more refined with the help of limit elements (a generalization of ω) until reaching a covering graph that is sufficiently precise to prove the absence of states covered by a given target. A sequence of under and over-approximations is used either to detect error traces or to prove correctness of the considered model. Completeness of the algorithm is non constructive, i.e., it is not possible to compute a priori the number of refinement steps needed for termination.

In [36] the authors introduce Petri nets with discrete parameters as a way to reason about families of concurrent systems. Instantiations of parameters give rise to standard Petri nets. Decision problems such as the existence of instantiations that can verify a given property or the synthesis of nets that can satisfy a parametric condition are considered in the paper. A general undecidability result is given for the considered decision problems. Syntactic fragments are identified in order to obtain decidability of properties like coverability with discrete parameters.

An alternative way to prove properties of Petri net models and, more generally of parameterized systems, consists in using abstractions and simulation relations in order to reduce verification problems to finite-state verification. These methods typically provide cut-off theorems for verification of temporal properties with quantification over process indexes. Abstractions and finite model theorems have been studied, e.g., in [33,93,34, 20,56,58,91] and used as parts of more complex analysis, e.g., in combination of model checking tools. Approaches based on program transformations and constraints have been considered, e.g., in [68]. In the rest of the paper we will focus our attention on parameterized models for systems with broadcast communication mechanisms.

3 Broadcast Protocols

In [57] Emerson and Namjoshi introduced a formal model of concurrent systems with broadcast communication called Broadcast Protocols. The model is based on an extension of German and Sistla's communicating automata. Specifically, a Broadcast Protocol is a tuple $\mathcal{P} = \langle Q, \Sigma, R, Q_0 \rangle$, where Q is a finite set of control states, Σ is a finite alphabet,

$$R \subseteq Q \times (\{!!a, ??a, !a, ?a \mid a \in \Sigma\} \cup \{\tau\}) \times Q$$

is the transition relation, and $Q_0 \subseteq Q$ is a set of initial control states. The label $!a$ (resp. $?a$) represents a rendez-vous between two processes on message $a \in \Sigma$ (a zero-capacity channel). The label $!!a$ (resp. $??a$) represents the capability of broadcasting (resp. receiving) a message $a \in \Sigma$. Given a process $\mathcal{P} = \langle Q, \Sigma, R, Q_0 \rangle$, a configuration with n processes is a tuple $\gamma = \langle q_0, \ldots, q_n \rangle$ of control states such that q_i is the current state of the i-the process for $i : 1, \ldots, n$. We use Γ (resp. Γ_0) to denote the set of configurations (resp. initial configurations) associated to $\mathcal{P}$. Note that even if Q_0 is finite, there are infinitely many possible initial configurations. For $q \in Q$ and $a \in \Sigma$, we define the set $R_a(q) = \{q' \in Q \mid \langle q, ??a, q' \rangle \in R\}$ which contains the states that can be reached from the state q when receiving the message a. We assume that $R_a(q)$ is non empty for every a and q, i.e., nodes always react to broadcast messages. Given a process $\mathcal{P} = \langle Q, \Sigma, R, Q_0 \rangle$, a Broadcast Protocol is defined by the transition system $\langle \Gamma, \rightarrow, \Gamma_0 \rangle$ where the transition relation $\rightarrow \subseteq \Gamma \times \Gamma$ is such that: for $\gamma, \gamma' \in \Gamma$ with $\gamma = \langle q_0, \ldots, q_n \rangle$, we have $\gamma \rightarrow \gamma'$ iff $\gamma' = \langle q'_0, \ldots, q'_n \rangle$ and one of the following condition holds

- $\exists i$ s.t. $\langle q_i, !!a, q'_i \rangle \in R$ and $q'_j \in R_a(q_j)$ for every $j \neq i$.
- $\exists i \neq j$ s.t. $\langle q_i, !a, q'_i \rangle, \langle q_j, ?a, q'_j \rangle \in R$ and $q'_l = q_l$ for every $l \neq i, j$.
- $\exists i$ s.t. $\langle q_i, \tau, q'_i \rangle \in R$ and $q'_j = q_j$ for every $j \neq i$.

For a Broadcast Protocol $\mathcal{P}$, the Coverability problem (control state reachability) consists in checking whether

there exists $n \geqslant 0$ and an initial configuration $\gamma = \langle q_0, \ldots, q_0 \rangle$ with n processes that can reach in one or more steps a configuration containing a certain state q.

By passing through a generalization of VASS, in [57] the authors define a forward procedure with acceleration that generalizes the Karp-Miller construction to Broadcast Protocols. In this setting we need transitions defined by the following equations

$$\mathbf{x}' = M \cdot \mathbf{x} + \boldsymbol{\delta}$$

where M is a matrix in which each column has a single 1, a property called unimodularity. The forward procedure operates by computing the effect of iterating the applications of transitions by extrapolating the limit operation of the resulting set of states. The procedure however is not guaranteed to terminate.

A decision procedure for Broadcast Protocols has been proposed by Esparza, Finkel and Mayr in [59]. The property is based on the observation that Broadcast Protocols form a wsts with respect to marking inclusion (i.e. point-wise ordering over vectors or natural numbers). Coverability for Broadcast Protocols can then be decided using a symbolic backward exploration algorithm that is an instance of the general algorithm for wsts [59]. As for Petri nets, constraints over integer variables can be used to symbolically explore the behavior of a protocol with an arbitrary number of processes. This property together with relaxations from integer to real arithmetic can be used to apply efficient constraint solvers for designing backward reachability engines [40, 38].

Broadcast Protocols have been used to model cache coherence protocols [38] (e.g. to model invalidation signals) and to model synchronization primitives (e.g. notifyAll in Java) for Boolean abstractions of concurrent programs [43].

3.1 Affine Well-structured Nets

An *affine well-structured net* (AWN) N [65] is given by a set of n places and a set of transitions. Each transition t is equipped with two n-vectors, F_t and H_t, and an $n \times n$-matrix G_t. A transition t can be fired whenever $F_t \leqslant M$. The effect of firing of transition t is defined by the equation $M' = (G_t \cdot (M - F_t)) + H_t$. The matrix G_t can be used to define whole place operations like reset and transfer arcs. For instance, if the i-th column of G_t is null, then the effect of G_t is to reset the contents of the i-th place. Actually, affine transformations can be applied to define effects obtained as linear combinations of the current number of tokens in the net. For instance, for a Petri net with n places, if the ith-place contain k_i tokens, then we can add $c_j k_i$ tokens in the j-th place or any linear combination obtained by combining $k_1, \ldots, k_n$ using fixed constants $c_1, \ldots, c_n$. AWN is a wsts model w.r.t. point-wise ordering of tuples of natural numbers. Therefore, coverability is still decidable in this model. AWN subsume models like VASS, Petri nets, Petri nets with reset and transfer arcs. Furthermore, AWN is strictly more expressive than Petri nets. Indeed, it has been proved that coverability is Ackermann-hard for Petri nets with reset or transfer arcs [100].

4 Broadcast Protocols with Data

In [30,37,39] we introduced a data-sensitive model of concurrent and distributed systems, called Multi Set Rewriting with Constraint (MSR($\mathcal{C}$), to extend the Broadcast Protocols of [57,59] in order to model data stored in individual processes. Indeed, the model combines multiset rewriting over atomic formulas with constraints and global operations expressed using a special kind of reaction rules. Formally, a MSR($\mathcal{C}$) specification is defined as follows. A constraint system $\mathcal{C}$ is defined by formulas with free variables in V, an interpretation domain $\mathcal{D}$, and a satisfiability relation $\models$ for formulas in $\mathcal{C}$ interpreted over $\mathcal{D}$. We use $\mathcal{D} \models_\sigma \varphi$ to denote satisfiability of φ via a substitution $\sigma : Var(\varphi) \to \mathcal{D}$, where $Var(\varphi)$ is the set of free variables in φ.

For a fixed set of predicates P, an atomic formula with variables has the form $p(x_1, \ldots, x_n)$ where $p \in P$ and $x_1, \ldots, x_n \in V$. A rewriting rule has the form $M \rightsquigarrow M' : \varphi$, where M and M' are multiset of atomic formulas with variables over P and V, and φ is a constraint formula over variables in $Var(M \oplus M')$ occurring in $M \oplus M'$. We use $M = A_1, \ldots, A_n$ to denote a multiset of atoms.

MSR(Id) is the instance obtained by considering the constraint system Id defined as follows. Constraint formulas are either conjunctions φ_1, φ_2 or atomic formulas of the form $x = y$ and $x < y$ for variables $x, y \in V$. The interpretation domain is defined over an infinite and ordered set of identifiers equipped with equality, namely $\langle Id, =, < \rangle$. For substitution $\sigma : V \to Id$, $x = y$ is interpreted as $\sigma(x) = \sigma(y)$, $x < y$ is interpreted as $\sigma(x) < \sigma(y)$, and φ_1, φ_2 is interpreted as $\sigma(\varphi_1) \wedge \sigma(\varphi_2)$. A constraint φ is satisfied by a substitution σ if $\sigma(\varphi)$ evaluates to *true*. A ground instance $M\sigma \rightsquigarrow M'\sigma$ of a rule $M \rightsquigarrow M' : \varphi$ is defined by taking a substitution $\sigma : Var(M \oplus M') \to Id$ such that $\sigma(\varphi)$ is satisfied in the interpretation Id.

As an example, consider the rule

$$p(x, y), q(x) \rightsquigarrow p(x, y), q(x), q(u) : x < u$$

The intuition is that processes $p(a,b)$ and $q(c)$ synchronize when $a = c$ and generate a new instance $q(d)$ with $c < d$. By associating natural numbers to identifiers, $p(1,2), q(1) \rightarrow p(1,2), q(1), q(4)$ and $p(3,10), q(3) \rightarrow p(3,10), q(3), q(8)$ are two instances of the considered rule. We use $Inst(R)$ to indicate the infinite set of instances of a set R of rules.

A configuration is a multiset N of atoms of the form $p(d_1, \ldots, d_n)$ with $d_i \in Id$ for $i : 1, \ldots, n$. For a set R of rules and a configuration N, a rewriting step is defined by the relation $\rightarrow$ s.t. $N = (M \oplus Q) \rightarrow (M' \oplus Q) = N'$ for $(M \rightsquigarrow M') \in Inst(R)$. A computation is a sequence of configurations $N_1 \ldots N_m \ldots$ s.t. $N_i \rightarrow N_{i+1}$ for $i \geqslant 0$.

Broadcast communication with data can be modeled in a direct way by adding a set of reaction rules to MSR($\mathcal{C}$) transitions [30]. Reactions play the role of receptions in Broadcast Protocols and are defined via rewriting rules of the form $p_1(\mathbf{u}) \rightharpoonup p_2(\mathbf{u}')$ with constraints defined on variables of both action and reaction rules. For instance, the rule

$$p_1(x) \rightsquigarrow p_2(x)$$
$$[q_1(y) \rightharpoonup q_2(y), q_1(z) \rightharpoonup q_3(z)] : y > x, x > z$$

specifies that when a process in state p_1 with data x moves to p_2, all processes in state q_1 with data larger than x move to q_2, whereas all processes in state q_1 with data smaller than x move to state q_3.

Coverability here is defined in terms of control state reachability, i.e., decide whether it is possible to reach a configurations that contains a given predicate symbol. This problem is decidable for specification in which predicates have at most arity one. The problem can be formulated as a more standard coverability problem by an adequate ordering of configurations that we will define after introducing variations of the model like CMRS.

Constrained Multiset Rewriting Systems (CMRS) [1] are an instance of multiset rewriting with constraints defined on top of a class of constraints over integers called gap-order constraints. Gap-order constraints have been introduced by Revesz [96] as an extension of Datalog. They are defined by conjunctions of formulas that are either equalities of the form $x = y$ or inequalities of the form $x + c \leqslant y$ where c is a natural number and x, y are variables over integer numbers. These constrains are less powerful than difference constraints (they are not closed under negation).

In [1] we have applied the theory of wsts to show that coverability is decidable for monadic CMRS. To define an ordering on configurations, we can proceed as follows. We first cluster predicates that have the same piece of data. A cluster is a multiset of predicate symbols. The resulting multisets can be ordered according to the relative order of data (we can consider all possible linearizations to deal with partial ordering induced by the guards). We then require the existence of a monotone injection between data occurring in two different markings so that the associated multiset of symbols are one included in the other. In other words markings can be viewed as words of multisets of symbols in P and the above mentioned ordering is a word embedding built on top of multiset inclusion. From Higman's lemma, the resulting ordering is a wqo. CMRS are monotone with respect to this ordering and predecessors of upward closed sets can be computed symbolically. The complexity of coverability for CMRS is non elementary, since CMRS subsumes reset and transfer nets.

Complexity results for verification of systems composed by a finite number of counters with transitions defined by gap-order constraints have been considered in [32]. CMRS differs from the models in [96,32] in that transitions are not restricted to a fixed number of counters. Furthermore, since constraints allow the use of inequalities, it is possible to model fresh name generation. For instance, starting from the configuration containing $p(0)$, the rule $p(x) \rightsquigarrow p(z), r(z) : z > x$, when repeatedly applied, injects an arbitrary number of atoms of the form $p(v)$ with increasing values as arguments. For instance, we have computations like $p(0) \rightarrow p(1), r(1) \rightarrow p(3), r(3), r(1) \ldots$. Applications of the resulting model inclused specification and analysis of (time-sensitive) cryptographic protocols for multiple sessions [31,41] and distributed protocols for mutual exclusion [8].

4.1 νNets

In [97,98] Velardo and De Frutos-Escrig introduced another model with data called ν-PN. ν-PN are an extension of Petri nets in which tokens are pure names that can only be compared for equality. To formally define the model, we consider a set Id of names, a set Var of variables and a subset of special variables $\Upsilon \subset Var$ used for fresh name creation. A *labelled ν-Petri Net* (ν-PN) is a tuple $N = (P, T, F, \lambda)$, where P and T are finite disjoint sets that represent places and transitions, respectively, $\lambda : T \rightarrow \Sigma_\epsilon$ is the labelling of transitions, and

$$F : (P \times T) \cup (T \times P) \rightarrow Var^{\oplus}$$

is such that for every $t \in T$, $pre(t) \cap \Upsilon = \varnothing$ and $post(t) \backslash \Upsilon \subseteq pre(t)$, where $pre(t) = \bigcup_{p \in P} supp(F(p,t))$ and $post(t) = \bigcup_{p \in P} supp(F(t,p))$ where $supp(S)$ is the set of names occurring in S. We also take $Var(t) = pre(t) \cup post(t)$. The mapping F labels every pair (p,t) and (t,p) by a multiset of variables. These variables

specify how tokens move from preconditions to post-conditions. Variables in Υ can only be instantiated to names that do not occur in the current marking. These special variables only appear in post-arcs.

As an example if variable x occurs as a label of (p,t), (r,t) and (t,q) then a token, with value d, can be removed resp. from place p and r and moved to place q without changing value. A similar transition can be expressed in MSR(Id) via a single rule of the form $p(x), r(x) \rightsquigarrow q(x) : true$. In the same example if variable $y \in \Upsilon$ labels (t,p), then a token with fresh value is added to place p. A similar transition can be expressed in MSR(Id) via a single rule of the form $p(x), r(x), max(z) \rightsquigarrow q(x), p(y), max(y) : y > z$ assuming that max always stores the highest value seen so far.

A marking of a ν-PN $N = (P, T, F, \lambda)$ is a mapping $M : P \to Id^{\oplus}$. We take $Id(M) = \bigcup_{p \in P} supp(M(p))$, the set of names in M. Thus, a marking M assigns to each place a multiset of names. Given a transition $t \in T$, a mode of t is a mapping $\sigma : Var(t) \to Id$ such that $\sigma(\nu_1) \neq \sigma(\nu_2)$ for each different $\nu_1, \nu_2 \in \Upsilon$. A transition t is *enabled* with mode σ for a marking M if for all $p \in P$, $\sigma(F(p,t)) \subseteq M(p)$ and $\sigma(\nu) \notin Id(M)$ for all $\nu \in \Upsilon$. Then t can be *fired* with mode σ, reaching the marking M' given by $M'(p) = (M(p) - \sigma(F(p,t))) + \sigma(F(t,p))$ for all $p \in P$. νNets turn out to be wsts with respect to an ordering similar to the wqo used for CMRS. Since data are not ordered we can relax the conditions on the embedding between data of different configurations and remove the monotonicity requirement. Coverability w.r.t. the above mentioned ordering is decidable in νNets [97].

4.1.1 Data Nets

Data nets are a generalization of CMRS and νNets in which tokens are colored with data taken from a infinite domain D equipped with a linear ordering $<$. A data net marking s is a multiset of tokens that carry data in D. We use $s(d)(p)$ to denote the number of tokens with data d in place p. An example of data net marking is as follows:

$$s = \left(\begin{array}{cc|cc|cc} \multicolumn{2}{c|}{v_1} & \multicolumn{2}{c|}{v_2} & \multicolumn{2}{c}{v_3} \\ p & q & p & q & p & q \\ 2 & 1 & 0 & 1 & 1 & 2 \end{array}\right)$$

where $s(v_1)(p)$ has two tokens, etc. Data nets transitions extend those of Petri nets by allowing deletion and addition of tokens with data. In addition they allow whole-place operations, e.g., transfer of all tokens with data in some region from one place to another region and place. For our purposes, it will be enough to consider transitions with arity $\alpha_t = 1$ defined below. For a non-deterministically chosen value d, a transition of arity 1 operates on the partitioning R_0, S_1, R_1 of the values contained in the current marking s.t. S_1 is the multiset of tokens with data $= d$, R_0 is the multiset of tokens with data $< d$, and R_1 is the multiset of tokens with data $> d$. A transition is defined by three matrices F_t, G_t, and H_t as in Fig. 1. First, for each place $p \in P$, $F_t(S_1, p)$ specifies how many tokens with data d have to be removed from place p. The transition is enabled iff there are enough tokens to remove from each place. G_t specifies transfers from one place to another that have to be executed after application of F_t. If $G_t(R_i, p, R_i, p) = 0$ (reset), and $G_t(R_i, p, S_1, q) = 1$ (transfer), then all tokens in place p with data in R_i are transferred to place q by updating their data into d (the only value in S_1). $G_t(R_i, p, R_i, q) = 1$ specifies a transfer from place p to place q within the same region (data remain unchanged), and $G_t(S_1, p, S_1, q) = 1$ has a similar effect on S_1. Finally, for each place $p \in P$, $H_t(S_1, p)$ specifies how many tokens with data d have to be added to place p. Consider the data net marking

$$F_t = \left(\begin{array}{cc|cc|cc} \multicolumn{2}{c|}{R_0} & \multicolumn{2}{c|}{S_1} & \multicolumn{2}{c}{R_1} \\ p & q & p & q & p & q \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right)$$

$$H_t = \left(\begin{array}{cc|cc|cc} \multicolumn{2}{c|}{R_0} & \multicolumn{2}{c|}{S_1} & \multicolumn{2}{c}{R_1} \\ p & q & p & q & p & q \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right)$$

$$G_t = \left(\begin{array}{cc|cc|cc|cc} & & \multicolumn{2}{c|}{R_0} & \multicolumn{2}{c|}{S_1} & \multicolumn{2}{c}{R_1} \\ & & p & q & p & q & p & q \\ \hline R_0 & p & 0 & 0 & 1 & 0 & 0 & 0 \\ & q & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline S_1 & p & 0 & 0 & 1 & 0 & 0 & 0 \\ & q & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline R_1 & p & 0 & 0 & 0 & 0 & 1 & 0 \\ & q & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right)$$

Fig. 1 A data net transition t with arity 1.

and the rule t with arity 1 defined in Fig. 1. We non-deterministically associate e_3 to S_1. Then, transition t first removes one token with value e_3 from place p. This produces the intermediate marking

$$s = \left(\begin{array}{cc|cc|cc|cc} \multicolumn{2}{c|}{e_1} & \multicolumn{2}{c|}{e_2} & \multicolumn{2}{c|}{e_3} & \multicolumn{2}{c}{e_4} \\ p & q & p & q & p & q & p & q \\ 2 & 1 & 2 & 1 & 2 & 3 & 2 & 2 \end{array}\right)$$

$$s_1 = \left(\begin{array}{cc|cc|cc|cc} \multicolumn{2}{c|}{e_1} & \multicolumn{2}{c|}{e_2} & \multicolumn{2}{c|}{e_3} & \multicolumn{2}{c}{e_4} \\ p & q & p & q & p & q & p & q \\ 2 & 1 & \mathbf{1} & 1 & 2 & 3 & 2 & 2 \end{array}\right)$$

We now apply the transfer operations, i.e, we reset region R_0 and place p and region R_1 and place q and and move all those tokens to region S_1 and place p. This

gives the intermediate configuration:

$$s_2 = \left(\begin{array}{cc|cc|cc|cc} & e_1 & & e_2 & & e_3 & & e_4 \\ p & q & p & q & p & q & p & q \\ \mathbf{0} & 1 & \mathbf{0} & 1 & \mathbf{7} & 3 & 2 & \mathbf{0} \end{array}\right)$$

Finally, we apply the addition step that yields the successor configuration:

$$s' = \left(\begin{array}{cc|cc|cc|cc} & e_1 & & e_2 & & e_3 & & e_4 \\ p & q & p & q & p & q & p & q \\ 0 & 1 & 0 & 1 & 7 & \mathbf{4} & 2 & 0 \end{array}\right)$$

The formal semantics of firings of data net transitions is given in [84]. We use $s \rightarrow_t s'$ to indicate the firing of a transition t from marking s to marking s' and with $\rightarrow$ the union of the relations $\rightarrow_t$ for each transition t.

The Coverability problem is defined on top of the following ordering between markings. Let $Data(s)$ be the set of data values that occur in a marking s. Then, $s_1 \preceq s_2$ iff there exists an injective function $h : Data(s_1) \mapsto Data(s_2)$ such that (i) h is monotone and (ii) $s_1(d)(p) \leqslant s_2(h(d))(p)$ for each $d \in Data(s_1)$ and $p \in P$. For instance, we have that

$$s = \left(\begin{array}{cc|cc|cc} & v_1 & & v_2 & & v_3 \\ p & q & p & q & p & q \\ 2 & 1 & 0 & 1 & 1 & 2 \end{array}\right)$$
$$\preceq$$
$$s' = \left(\begin{array}{cc|cc|cc|cc} & u_1 & & u_2 & & u_3 & & u_4 \\ p & q & p & q & p & q & p & q \\ 3 & 2 & 1 & 2 & 0 & 1 & 2 & 4 \end{array}\right)$$

via the injection h that maps v_1 to u_1, v_2 to u_2 and v_3 to u_4. For markings s_0 and s_1, the data nets coverability problem consists in checking if there exists a marking s_2 s.t. $s_0 \rightarrow^* s_2$ and $s_1 \preceq s_2$. Data nets turn out to be wsts. Therefore, coverability is decidable for Data nets [84].

4.2 Expressiveness

In [6,45] we studied the relative expressive power of infinite-state models like Petri nets, Broadcast Protocols, AWN, νNets, MSR(Id)/CMRS, and Data Nets. An adequate measure to compare the expressive power of all these models is based on the use of languages associated to computations. More specifically, assume that each transition has a label taken from a finite alphabet Σ. Furthermore, assume that the label of the transition is used as a label of the corresponding transitions between (global) states. A computation then generates a word over Σ^*. The language associated to a model depends on the conditions used to accept a word. Since the goal is to compare the expressive power of wsts models, we consider accepting conditions defined as in the coverability problem. Given a labelled transition system $S = (S, \Sigma, \rightarrow, s_0, s_f)$ with a quasi-ordering $\leqslant$, we define the covering language:

$$\mathcal{L}(S) = \{w \in \Sigma^* \mid s_0 \xrightarrow{w} s,\ s_f \leqslant s\}$$

The covering language is strictly related to the coverability. It captures the language generated by computations that lead to a state that is larger than a fixed target state s_f. Given a model M, we use $\mathcal{L}(M)$ to denote the class of languages defined by instances in M. For instance, for the class of finite automata FA, $\mathcal{L}(FA)$ is the set of regular languages. By comparing the class of generated languages, we can study the relative expressive power of different types of wsts. From the results presented in [64,71,6,45], we have the following strict hierarchy between the wsts models described in the previous sections:

$$\mathcal{L}(FA) \subset \mathcal{L}(PN) \subset \mathcal{L}(AWN) \subset$$
$$\subset \mathcal{L}(\nu Nets) \subset \mathcal{L}(DN)$$

where FA, PN, DN stand for finite automata, Petri nets, and Data nets respectively. The strict inclusion $\mathcal{L}(\nu{-}Nets) \subset \mathcal{L}(DN)$ has been proved using an application of ordinal theory [29] as a measure of the state space that has to be explored to solve coverability in the corresponding models. The method used in [29] based on order reflection is a generalization of the strict inclusion between Lossy Channel Systems and CMRS proved in [6].

Furthermore, in [6] it has been shown that

$$\mathcal{L}(DN) = \mathcal{L}(CMRS) = \mathcal{L}(MSR(Id))$$

The latter result is quite surprising. It shows that, when considering languages accepted with coverability, broadcast communication increase the expressive power of Petri nets, whereas whole place operations (a generalization of broadcast communication) do not increase the power of Petri nets with ordered data, i.e., Data Nets are equivalent to MSR(Id) and CMRS. The intuition behind the latter result is that by using ordered data it is possible to simulate a broadcast operation via a sequence of stages identified by increasing values. Broadcast is encoded then as a sequence of individual transfers of tokens from one stage to the next one. This transfer can lose some of the token. However this is not a problem when considering coverability problems. Indeed it can be shown that coverability in a lossy version of a wsts model is equivalent to coverability in the non-lossy version.

We remark that AWN are a generalization of Broadcast Protocols, BP for short, but from the analysis in [64], we also have that

$$\mathcal{L}(PN) \subset \mathcal{L}(BP)$$

Finer grained comparisons between AWN, νNets and (fragments of) Data nets are presented in [29]. More general models of Petri nets with data are presented in [89].

4.3 Other Models and Verification Methods

Parameterized concurrent programs communicating via a shared memory have been considered, e.g., in [22] as abstract models of multithreaded programs. In [60] Esparza, Ganty and Majumdar study the complexity of systems containing a leader process and arbitrarily many anonymous and identical contributors. Communication is defined via atomic operations on a shared register. In [8,52] the authors presented approximated procedures for concurrent systems combining broadcast communication and universally quantified conditions. The key ingredient is the application of Monotonic Abstraction [18] an abstraction that transforms pre-conditions into post-conditions in order to enforce monotonicity. Similar approximations have been considered in symbolic backward engines based on SMT solvers [19]. SMT and SAT solvers can also been applied to solve coverability for Petri nets [61], and for well-structured transition systems [81]. Techniques for automatically constructing counting arguments, by synthesizing counter predicates, for programs with infinite control have been developed in [62,63].

5 Distributed Broadcast Protocols

In this section we focus our attention on graph-based models with broadcast communication, an extension of Broadcast Protocols.

In [48] we introduced an extension of Broadcast Protocols, called Ad Hoc Networks (AHN), in which processes are distributed on a graph. To formalize this idea, let us first define a Q-graph as a labeled undirected graph $\gamma = \langle V, E, L\rangle$, where V is a finite set of *nodes*, $E \subseteq V \times V \backslash \{\langle v, v\rangle \mid v \in V\}$ is a finite set of *edges*, and L is a labeling function from V to a set of labels Q. We use $L(\gamma)$ to represent all the labels present in γ. The nodes belonging to an edge are called the *endpoints* of the edge. A process is a tuple $\mathcal{P} = \langle Q, \Sigma, R, Q_0\rangle$, where Q is a finite set of control states, Σ is a finite alphabet, $R \subseteq Q \times (\{!!a, ??a \mid a \in \Sigma\} \cup \{\tau\}) \times Q$ is the transition relation, and $Q_0 \subseteq Q$ is a set of initial control states. The label $!!a$ (resp. $??a$) represents the capability of broadcasting (resp. receiving) a message $a \in \Sigma$. As for Broadcast Protocols, we define $R_a(q) = \{q' \in Q \mid \langle q, ??a, q'\rangle \in R\}$ as the set of states that can be reached from the state q when receiving the message a and assume that $R_a(q)$ is non empty for every a and q. We also consider local transitions of the form $\langle q, \tau, q'\rangle$. We do no consider here rendez-vous communication. Given a process $\mathcal{P} = \langle Q, \Sigma, R, Q_0\rangle$, a configuration is a Q-graph and an initial configuration is a Q_0-graph. We use Γ (resp. Γ_0) to denote the set of configurations (resp. initial configurations) associated to $\mathcal{P}$. Note that even if Q_0 is finite, there are infinitely many possible initial configurations (the number of Q_0-graphs). Communication is achieved via selective broadcast, which means that a broadcast message is received by the nodes which are adjacent to the sender. Non-determinism in reception is modeled by means of graph reconfigurations. We next formalize this intuition. Given a process $\mathcal{P} = \langle Q, \Sigma, R, Q_0\rangle$, an AHN is defined by the transition system $\langle \Gamma, \rightarrow, \Gamma_0\rangle$ where the transition relation $\rightarrow \subseteq \Gamma \times \Gamma$ is such that: for $\gamma, \gamma' \in \Gamma$ with $\gamma = \langle V, E, L\rangle$, we have $\gamma \rightarrow \gamma'$ iff $\gamma' = \langle V, E, L'\rangle$ and one of the following condition holds

- $\exists v \in V$ s.t. $\langle L(v), !!a, L'(v)\rangle \in R$ and $L'(u) \in R_a(L(u))$ for every $\langle u, v\rangle \in E$, and $L(w) = L'(w)$ for any other node w.
- $\exists v \in V$ s.t. $\langle L(v), \tau, L'(v)\rangle \in R$, and $L(w) = L'(w)$ for any other node w.

The model is inspired by graph-based models of distributed systems presented in [76,99,102,103].

Dynamic network reconfigurations can be modeled by adding transitions in which the set of edges is nondeterministically changed. Hence, we extend the transition relation in order to include the following case: $\gamma, \gamma' \in \Gamma$ with $\gamma = \langle V, E, L\rangle$, we have $\gamma \rightarrow \gamma'$ if $\gamma' = \langle V, E', L\rangle$ for some $E' \subseteq V \times V \backslash \{\langle v, v\rangle \mid v \in V\}$.

Parameterized verification problems for our model can be defined by considering the following type of reachability queries. Given a process $\mathcal{P}$, a transition system $\langle \Gamma, \rightarrow, \Gamma_0\rangle$, and a control state q, the coverability problem consists in checking whether or not there exists $\gamma_0 \in \Gamma_0$ and $\gamma_1 \in \Gamma$ s.t. $\gamma_0 \rightarrow^* \gamma_1$ and $q \in L(\gamma_1)$. We remark that the initial configuration is not fixed a priori. In fact, the only constraint that we put on the initial configuration is that the nodes have labels taken from Q_0 without any information on their number or connection links. Similar problems can be studied for the variations of the basic model with the following features: node crashes, asynchronous communication, messages with data fields and nodes with local memory.

According to our semantics, the number of nodes stays constant in each execution starting from the same

initial configuration. As a consequence, when fixing the initial configuration γ_0, we obtain finitely many possible reachable configurations. Checking the parameterized version of the reachability problem is generally much more difficult. The problem easily gets undecidable since for parametric initial configurations we have to deal with an infinite family of transition systems (one for each initial graph). Despite of it, we can still find interesting restrictions to the model or to the set of considered configurations for which coverability is decidable.

5.1 Synchronous Broadcast

For this model, we have proved in [48] that coverability is undecidable without dynamic reconfiguration, i.e., when the topology never changes during execution. The proof is based on a discovery protocol implemented by running the same process on each node of the network (whose shape is unknown). The discovery protocol controls interferences by forcing states receiving more copies of the same message into special dead states. This strategy can then be used to navigate into unknown networks and to select one by one nodes that belong to a subgraph with a given shape. A simple shape like a list of fixed but arbitrary length is enough to run a simulation of a Two Counter machine (the list models the maximal aggregate value of the counters). Interestingly, the problem becomes decidable when considering non-deterministic reconfiguration steps. Non-deterministic reconfiguration steps destroy the influence of the topology on the behavior of individual nodes. The results can then be proved via a reduction to coverability for Petri nets. A broadcast is simulated via a rendez vous with an arbitrary subset of nodes in the network. It is interesting to remark that, as in other models like channel systems, see, e.g., [14], the loss of information has the effect of simplifying the verification task (the state space becomes more regular) while it complicates the design of protocols (programming in the model is harder).

To emphasize this point, in [47] we have shown that, in presence of non-deterministic reconfigurations, the decision procedure for coverability has polynomial time complexity in the size of the input protocol. The proof is based on a labeling algorithm that exploits monotonicity properties of the semantics with reconfigurations steps. The algorithm can be viewed as a saturation process that marks as visited every state that can be generated by via synchronization step starting from any number of copies of already visited states. The algorithm requires at worst as many step as the number of control states in the protocol. More complex parametric reachability properties in which the target configurations are generated by constraints on the number of occurrences of control states can still be decided but with increasing complexity. For instance, the problem becomes exponential when target states are described by conjunctions of interval constrains defined over occurrences of states [47].

5.2 Restricted Topologies

In [48,49] we have introduced a restricted form of coverability in which configurations are required to belong to a fixed subclass of graphs (e.g. stars, fully connected graphs, etc.). A quite interesting example of non trivial class consists of all undirected graphs in which the length of simple paths is bounded by the same constant k (k-bounded path graphs). For $k \geqslant 1$, we still have an infinite set of graphs (e.g. $k = 2$ contains all stars with diameter two). Notice that fully connected graphs are not bounded path.

For synchronous broadcast communication without reconfiguration, parameterized verification is still decidable for bounded path graphs. The results follows from a non trivial application of the theory of well-structured transition systems [5,66]. Our model is well-structured on the class of bounded path graphs with respect to the induced subgraph relation. Let us define the ordering starting from the usual subgraph relation. Given two graphs $G = \langle V, E, L\rangle$ and $G' = \langle V', E', L'\rangle$, G is in the *subgraph* relation with G', written $G \sqsubseteq_s G'$, whenever there exists an injection $f : V \to V'$ such that, for every $v, v' \in V$, if $\langle v, v'\rangle \in E$, then $\langle f(v), f(v')\rangle \in E'$.

The induced subgraph ordering has the following stronger requirements. Given two graphs $G = \langle V, E, L\rangle$ and $G' = \langle V', E', L'\rangle$, G is in the *induced subgraph* relation with G', written $G \sqsubseteq_i G'$, whenever there exists an injection $f : V \to V'$ such that, for every $v, v' \in V$, $\langle v, v'\rangle \in E$ if and only if $\langle f(v), f(v')\rangle \in E'$. The two orderings are not equivalent. As an example, a path with three nodes is a subgraph, but not an induced subgraph, of a ring of the same order. Subgraph and induced subgraph are well-quasi ordering for bounded path graphs a result due to Ding [54]. Broadcast communication is monotone with respect to induced subgraph but not with respect to subgraph. These properties can be used to obtain a well-structured transition systems for our extended notion of broadcast protocols over the set of bounded path configurations. The algorithm in [48] operates on finite representations of infinite set of configurations. The decidability result can be extended to a slightly more general class of graphs that includes both stars and cliques [49]. The bounded

path restriction is used in [49] on graphs obtained after collapsing cliques into single nodes. More precisely, the maximal clique graph K_G associated to a graph $G = (V, E, L)$ is the bipartite graph $\langle X, W, E', L' \rangle$ in which $X = V$, $W \subseteq 2^V$ is the set of maximal cliques of G, for $v \in V, w \in X$, $\langle v, w \rangle \in E'$ iff $v \in w$; $L'(v) = L(v)$ for $v \in V$, and $L'(w) = \bullet$ for $w \in W$. We reformulate the bounded path condition on the maximal clique graph associated to a configuration (i.e. the length of the simple paths of K_G is at most n). For $n \geqslant 1$, the class BPN_n consists of the set of configurations whose associate maximal clique graph has n-bounded paths (i.e. the length of the simple paths of K_G is at most n). The ordering we are interested in is defined on maximal clique graphs as follows. Assume $G_1 = \langle V_1, E_1, L_1 \rangle$ with $K_{G_1} = \langle X_1, W_1, E_1', L_1' \rangle$, and $G_2 = \langle V_2, E_2, L_2 \rangle$ with $K_{G_2} = \langle X_2, W_2, E_2', L_2' \rangle$ with G_1 and G_2 both connected graphs. Then, $G_1 \sqsubseteq_m G_2$ iff there exist an injection $f : X_1 \to X_2$ and $g : W_1 \to W_2$, such that

1. for every $v \in X_1$, and $C \in W_1$, $v \in C$ iff $f(v) \in g(C)$;
2. for every $v_1, v_2 \in X_1$, and $C \in W_2$, if $f(v_1) \sim_C f(v_2)$, then there exists $C' \in W_1$ s.t. $f(v_1) \sim_{C'} f(v_2)$;
3. for every $v \in X_1$, $L(v) = L(f(v))$;
4. for every $C \in W_1$, $L(C) = L(g(C))$.

It holds that $G_1 \sqsubseteq_m G_2$ iff $G_1 \sqsubseteq_i G_2$ (G_1 is an induced subgraph of G_2). Furthermore, the resulting ordering is still a wqo and the transition relation $\to$ is still monotone.

5.3 Faults and Conflicts

In [50] we have studied the impact of node and communication failures on the coverability problem for our model of ad hoc network protocols. We started our analysis by introducing node failures via an intermittent semantics in which a node can be (de)activated at any time. Coverability is decidable under the intermittent semantics. Decidability derives from the assumption that nodes cannot take decisions that depend on the current activation state (e.g. change state when the node is turned on). We then consider two restricted types of node failure, i.e., node crash (a node can only be deactivated) and node restart (when it is activated, it restarts in a special restart state). Coverability becomes undecidable in these two semantics. We considered then different types of communication failures. We first consider a semantics in which a broadcast is not guaranteed to reach all neighbors of the emitter nodes (message loss). Coverability is again decidable in this case. We then introduce a semantics for selective broadcast specifically designed to capture possible conflicts during a transmission. A transmission of a broadcast message is split into two different phases: a starting and an ending phase. During the starting phase, receivers connected to the emitter move to a transient state. While being in the transient state, a reception from another node generates a conflict. In the ending phase an emitter always moves to the next state whereas connected receivers move to their next state only when no conflicts have been detected. In our model we also allow several emitters to simultaneously start a transmission. Decidability holds only when receivers ignore corrupted messages by remaining in their original state. Moreover, in all cases the above mentioned models in which coverability is decidable the decision procedure can be defined via polynomial time reachability algorithm similar to that used in the case of reconfigurations.

5.4 Time

In [10] we have considered a timed version of AHN in which each node has a finite number of dense/discrete clocks. Time elapsing transitions increase all clocks at the same rate. The resulting model extends the Timed Networks model of [17] with an underlying connection graph. When constraining communication via a complex connection graph, the decidability frontier becomes much more complex. For nodes equipped with a single clock, coverability is already undecidable for graphs in which nodes are connected so as to form stars with diameter five. The undecidability result can be extended to the more general class of graphs with bounded simple path (for some bound $N \geqslant 5$ on the length of paths). We remark that in the untimed case coverability is decidable for bounded path topologies and stars. Coverability is undecidable for fully connected topologies in which each timed automaton has at least two clocks. Decidability holds for special topologies like stars with diameter three and fully connected graphs if nodes have at most one clock. For discrete time coverability is decidable for nodes with finitely many clocks for fully connected topologies and graphs with bounded path.

5.5 Asynchronous Broadcast

In [51] we have enriched the model in order to consider asynchronous communication implemented via mailboxes attached to individual nodes and consider different policies for handling mailboxes (unordered and fifo) and the potential loss of messages. In this model nodes have an additional local data structure that models the mailbox. Interestingly, even if the model is apparently richer than the synchronous one, coverability is still decidable

in some case. More specifically, we consider a mailbox structure $\mathbb{M} = \langle \mathcal{M}, del?, add, del, \flat \rangle$, where $\mathcal{M}$ is a denumerable set of elements denoting possible mailbox contents; $add(a, m)$ denotes the mailbox obtained by adding a to m, $del?(a, m)$ is true if a can be removed from m; $del(a, m)$ denotes the mailbox obtained by removing a from m when possible, undefined otherwise. Finally, $\flat \in \mathcal{M}$ denotes the empty mailbox. We call an element a of m *visible* when $del?(a, m) = true$. Their specific semantics and corresponding properties changes with the type of mailbox considered. A protocol is defined by a process $\mathcal{P} = \langle Q, \Sigma, R, q_0 \rangle$ as in the AHN model with the same notation for broadcast messages. Configurations are undirected $Q \times \mathcal{M}$-graphs. A $Q \times \mathcal{M}$-graph γ is a tuple $\langle V, E, L \rangle$, where V is a finite set of nodes, $E \subseteq V \times V$ is a finite set of edges, and $L : V \to Q \times \mathcal{M}$ is a labeling function. $\mathcal{C}_0$ is the set of undirected graphs in which every node has the same label $\langle q_0, \flat \rangle$ that denotes the initial state of individual processes. Given the labeling L and the node v s.t. $L(v) = \langle q, m \rangle$, we define $L_s(v) = q$ (state component of $L(v)$) and $L_b(v) = m$ (buffer component of $L(v)$). Furthermore, for $\gamma \in \mathcal{C}$, we use $L_s(\gamma)$ to denote the union of the set of control states of nodes in γ ($L_s(\gamma) = \bigcup_{v \in V} L_s(v)$ for $\gamma = \langle V, E, L \rangle$). For $\mathbb{M} = \langle \mathcal{M}, del?, add, del, \flat \rangle$, an Asynchronous Broadcast Network (ABN) is defined by the transition system $\mathcal{T}(\mathcal{P}, \mathbb{M}) = \langle \mathcal{C}, \to, \mathcal{C}_0 \rangle$, where $\to \subseteq \mathcal{C} \times \mathcal{C}$ is the transition relation defined next.

For $\gamma = \langle V, E, L \rangle$ and $\gamma' = \langle V, E, L' \rangle$, $\gamma \to \gamma'$ holds iff one of the following conditions on L and L' holds:

- (local) there exists $v \in V$ such that $(L_s(v), \tau, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$, and $L(u) = L'(u)$ for each $u \in V \backslash \{v\}$.
- (broadcast) there exists $v \in V$ and $a \in \Sigma$ such that $(L_s(v), !!a, L'_s(v)) \in R$, $L_b(v) = L'_b(v)$ and for every $u \in V \backslash \{v\}$
  - if $\langle u, v \rangle \in E$ then $L'_b(u) = add(a, L_b(u))$ and $L_s(u) = L'_s(u)$,
  - otherwise $L(u) = L'(u)$;
- (receive) there exists $v \in V$ and $a \in \Sigma$ such that $(L_s(v), ??a, L'_s(v)) \in R$, $del?(a, L_b(v))$ is satisfied, $L'_b(v) = del(a, L_b(v))$, and $L(u) = L'(u)$ for each $u \in V \backslash \{v\}$.

Coverability for ABN is defined as in the case of AHN, i.e., we search for an initial configuration that via a finite number of steps can reach a configuration that exposes a given control state $q \in Q$. Decidability of coverability is strictly related to the policy used to handle mailboxes.

Multiset The mailbox structure *Bag* is defined as follows: $\mathcal{M}$ is the denumerable set of bags over Σ, $add(a, m) = [a] \oplus m$ (where $[a]$ is the singleton bag containing a), $del?(a, m) = true$ iff $m(a) > 0$, $del(a, m) = m \ominus [a]$, and $\flat \in \mathcal{M}$ is the empty bag $[]$.

When local buffers are treated as bags of messages the coverability problem is decidable. The proof is based on two steps. We can first show that, for the purpose of deciding coverability, we can restrict to fully connected topologies only. We can then use a reduction to the PTime-complete algorithm of [47].

FIFO The mailbox structure *FIFO* is defined as follows: $\mathcal{M}$ is defined as Σ^*; $add(a, m) = m \cdot a$ (concatenation of a and m); $del?(a, m) = true$ iff $m = a \cdot m'$; $del(a, m)$ is the bag m' whenever $m = a \cdot m'$, undefined otherwise; finally, $\flat \in \mathcal{M}$ is the empty string ϵ. When mailboxes are ordered buffers, we obtain undecidability already in the case of fully connected topologies. The coverability problem becomes decidable when introducing non-deterministic message losses. In an extended model in which a node can test if its mailbox is empty, we obtain undecidability with unordered bags and both arbitrary or fully-connected topologies.

5.6 Distributed Broadcast Protocols with Data

In [46] we considered a further refinement step by introducing local registers and data fields in message payloads. We model a distributed network using a graph in which the behavior of each node is described via an automaton with operations over a finite set of registers. A node can transmit part of its current data to adjacent nodes using broadcast messages. A message carries both a type and a finite tuple of data. Receivers can either test, store, or ignore the data contained inside a message. We assume that broadcasts and receptions are executed without delays (i.e. we simultaneously update the state of sender and receiver nodes).

Our analysis shows that, even in presence of register automata, dynamic reconfiguration can still render the coverability problem easier to solve. More precisely, in fully connected topologies coverability is undecidable for nodes with two registers and messages with one field. Coverability remains undecidable with dynamic network reconfigurations if nodes have two registers but messages have two fields. Decidability holds for $k \geqslant 1$ registers and a single data field per message for arbitrary topologies and dynamic network reconfiguration. The decision algorithm is based on a saturation procedure that operates on a graph-based symbolic representation of sets of configurations in which the data are abstracted away. This is inspired by similar techniques used in the case of classical register automata [79]. The problem is PSpace-complete in this case. Finally, for

fully connected topologies but without dynamic reconfiguration, coverability for nodes with a single register and messages with a single field is decidable with non elementary complexity. The decidability proof exploits the theory of well-structured transition systems [16,66]. The non-elementary lower bound follows from a reduction from coverability in reset nets [101].

5.7 Probabilistic Distributed Broadcast Protocols

Networks of probabilistic automata have been studied in [25,26,27]. The first probabilistic version of the AHN model is considered in [25]. Each node is described by a probabilistic single-clock automaton with broadcast communication. Parameterized verification is defined by requiring that at least one process in an error state is reached almost-surely under all scheduling policies. For static configurations the problem is undecidable as in the model without probabilities. Decidability however holds for dynamic reconfiguration of the network. In [26,27] the authors further refine their analysis by considering the existence of deterministic strategies and local strategies in which all nodes behave in the same way to reach a given state almost surely.

5.8 Directed Acyclic Topologies

A model of broadcast communication with topologies represented as acyclic directed graphs has been presented in [4]. In this setting broadcast communication is unidirectional since edges are directed. Coverability is defined with respect to subgraph ordering for directed graphs. The authors first show that coverability remains undecidable for directed acyclic graphs. The reduction is based on an encoding of the emptiness test for the transitive closure of a (regular) transducer relation, i.e., given two regular languages L and L' and a transducer relation R, decide whether $R^i(L) \cap L' \neq \emptyset$ for some i. The idea is similar to the encoding of counters used for AHN. An initial node non-deterministically decides to get the role of the automaton that accepts L. It then sends a notification to all successors and then non deterministically selects a word in L and broadcasts each of its symbols to all successors. A node that receives a notification gets the role of a copy of a transducer T accepting relation R and then start translating the incoming word w in $R(w)$ broadcasting each symbols to the successors. This can be repeated for an sequence of arbitrary length of nodes until reaching a nodes that non-deterministically gets the role of the automaton recognizing L'. The resulting construction forms a pipeline in which each node sends a letter to one successor. Interferences can be controlled by sending nodes that receive more than one notification to an error state. Since the graph is a directed and acyclic, the only interference are due to two or more conflicting incoming messages.

The authors then show that coverability for topology restricted to DAGs of bounded height is decidable with broadcast communication. The theory of wsts cannot be applied directly since the subgraph relation is not a wqo for DAGs of bounded height. However, given the particular type of systems considered here, in each node there is an automata than sends broadcast messages to successor nodes, it is possible to apply some transformation from DAGs of bounded height to Trees of bounded heights and then apply wsts theory on a new ordering defined on the resulting structures. The key point is that of encoding a DAG into an inverted tree by traversing a DAG backwards and splitting nodes to obtain a tree structure. Splitting nodes maintaining the same control state does not change the behavior of the model. If a node is split in two copies with the same state, then the two copies can send the same sequence of messages to successor nodes. The transformation may produce a forest. However each tree in the forest can be considered in isolation. Wsts theory is then applied to inverted trees and a new wqo ordering applied to show that coverability is decidable on inverted trees of bounded height. A generalization of the model in [4] has been introduced in [3]. In this setting links are implicitly defined by using identifiers (stored in local registers) as channel names. Dynamic modifications of the topology are controlled via updates to local registers. Coverability is undecidable in the resulting model even for bounded path topologies. Further restrictions, with effects similar to non deterministic reconfigurations of topologies, are needed in order to obtain decidability results for the model.

5.9 Other Graph-based Models

In [13] we apply graph-based transformations to model intermediate evaluations of non-atomic mutual exclusion protocols with universally quantified conditions. Parameterized verification is undecidable in the resulting model. Semi-decision procedures can be defined by resorting to upward closed abstractions during backward search (monotonic abstraction as in [8,52]). In [42] we studied decidability of reachability and coverability for a graph-based specification used to model biological systems called kappa-calculus [35]. Among other results, we proved undecidability for coverability for graph rewrite systems that can only increase the size of a configuration. Reachability problems for graph-based

representations of protocols have also been considered in [9] where symbolic representations combining a special graph ordering and constraint-based representation of relations between local data of different nodes have been used to verify parameterized consistency protocols.

To generalize some of the ideas studied in [42,9,52, 48,49], in [24] we have studied reachability and coverability (i.e., reachability of graphs containing specific patterns) for Graph Transformation Systems (GTS). Specifically, by transferring in the GTS setting the results in [48], in [24] we have shown that coverability is decidable for GTS for graph with bounded path graphs ordered via subgraph inclusion. The latter result follows from the theory of well-structured transition systems.

6 Tools for Parameterized Verification

Verification procedures for parameterized systems are often based on abstractions and heuristics in order to go beyond the limitations imposed by the undecidability or complexity results discussed in the previous sections. Tools like TRex, FAST, and LASH are devised for the larger class of counter systems (VAS, Petri nets, etc). Procedures like MCMT, MSR($\mathcal{C}$), and MAP are general purpose methods based on declarative specification languages like first order logic or constraint logic programming. Specialized engines like PFS, UNDIP, and BOOM exploit the structure of the model (e.g. automata/programs that specifies a single process/node) to obtain abstraction and heuristics for a global analysis. In the rest of the section we will briefly describe some of the existing prototypes that can handle models of parameterized systems that are close in spirit to Broadcast Protocols or their extensions.

- FAST [23,113] is a tool for forward analysis of counter systems defined via affine relations with parameters (i.e. it can handle VAS, Petri nets with reset and transfer arcs, etc). FAST is based on accelerations defined via Presburger formulas.
- TRex [112] is based on *Parametric Difference Bound Matrices*, an extension of Bounded Difference Matrices with external constraints to handle parameters, and accelerations defined on top of them.
- LASH [114] is a tool that can perform symbolic reachability analysis using automata as a symbolic representation of Presburger arithmetics, in other words automata accept the language that corresponds to encodings of vectors that satisfy a given formula. Automata can be used to define combination and transformation of constraints in a purely algorithmic way.
- MIST[105] is a tool based on abstraction refinement based on the Expand, Enlarge and Check approach [116] that exploits efficient representations of upward closed sets of states based on Interval Sharing Trees, an extension of the Covering Sharing Trees defined in [44].
- MSR($\mathcal{C}$) [109] is a tool based on Constraint Logic Programming that implements forward and backward symbolic reasoning for multiset rewriting with constraints. The analysis is based on the use of minimal representation (based on predicates with constraints) to represent infinite sets of configurations.
- MAP [67] is a tool based on transformations of constraint logic programs that can be applied to infinite-state systems with linear configurations and relations over data variables.
- MCMT [106] is a symbolic backward reachability engine based on SMT solvers that can handle parameterized systems with linear configurations. The MCMT tool is based on the EPR fragment of first order logic with arrays and applies different types of heuristics including invariant generation to reduce the state space.
- PFS [110] and UNDIP [111] are tool specifically devised to handle parameterized systems. They are both based on symbolic backward search but they target different types of systems. PFS deals with families of finite-state automata with global conditions whereas UNDIP can handle models of distributed systems with data [8,7,52]. The tools are based on monotonic abstractions, i.e., an abstraction that computes an over-approximation of the reachability set based on upward closed set of states.
- AUGUR 2 [82] is a tool devised for the analysis of Graph Transformation Systems using approximated unfoldings based on Petri nets. The approximated systems can then be verified using regular expressions, first order logic and coverability checking techniques. AUGUR 2 does not handle global operations like those needed for modeling broadcast communication.
- UNCOVER [104,115] is a tool that performs a symbolic backward reachability analysis for GTS with universally quantified conditions. The tool exploits a generalization of monotonic abstraction to quantifications over graph patterns as a heuristic to manipulate infinite sets of configurations using minimal constraints (given in form of graphs) only. UNCONVER can be viewed as the counterpart of UNDIP and PFS for systems in which configurations have a graph structure.
- PETRUCHIO [92] is a tool that extracts a Petri net representation from specifications of dynamic

networks based π-calculus. The tool discovers fragments of dynamic processes from more complex representations and translate them into Petri net places regulated by transitions that can be used then to explore the infinite state space of the original model. Termination is given for specifications that have a finite basis of fragments (i.e. they can be represented by a Petri net with finitely many places). A forward exploration procedure based on the Karp-Miller covering graph is used to reason on the behavior of the fragments extracted from the high level specification. Similar ideas have been applied to obtain decidability results for the analysis of fragments of (different types of) process algebra.

- Boom [108] is a tool that applies symbolic algorithms, see, e.g., [78,83,88,77], to verify counter abstractions of multithreaded programs. The algorithms behind the tool go beyond backward search. Indeed they combine several types of heuristics like those based on dynamic generation and refinement of over-approximations (defined in terms of upward closed set of states).
- PCW [107] is a tool that applies ordered counter abstraction [69], a refinement of monotonic abstraction with CEGAR, for the verification of parameterized systems. In this setting over-approximations are refined by using stronger and stronger orderings that can be used to define upward closed sets that "forbid" specific patterns (e.g. they forbid sets of points defined by a given equation).

Finally, we mention other existing methods that have experimental implementations that can handle interesting classes of concurrent and distributed systems. View Abstraction applies finite-state abstractions for verification of parameterized systems with global conditions and that has refinement produce that is complete for well structured transition systems [11,12]. The tool automatically detects cut-off points by performing a sort of iterative deepening on the abstract domain chosen for the considered systems (e.g. configurations with up to K processes). A prototype model checker for parameterized fault tolerant systems based on abstractions is described in [74]. Specifications are given here with an extension of Promela with parameters. An abstract domain based on intervals is used in order to apply counting abstractions on the original model.

7 Conclusions and Future Directions

In this paper we have given an overview of formal specification frameworks and verification techniques that can be applied to model concurrent and distributed systems with broadcast communication. The paper collects work done by the author in collaboration with several researchers as well as related results obtained on formal models of broadcast communication. The focus of the paper is on verification algorithms mainly from problems formulated over an infinite-state space. The presentation of the different models is given in a uniform way by using transition systems as a common language to describe their semantics. Communication in fully connected topologies or graph topologies can be used to express different classes of systems like multithreaded programs, cache coherence protocols and distributed algorithms. Most of the decidability results described in the paper have no practical counterpart in terms of verification tools or decision procedures. Finding classes of systems for which the considered model can be applied is an interesting direction for this kind of research. In view of the high complexity of the considered decision procedures, termination guarantees are often only of theoretical interest. The application of the verification procedures based on general frameworks like wsts to models that are not in decidable fragments can be an interesting direction for transferring the results to practical examples.

Acknowlegments The paper is based on previous work done in collaboration with the following researchers: P. A. Abdulla, M. F. Atig, N. Ben Henda, N. Bertrand, M. Bozzano, C. Di Giusto, J. Esparza, M. Gabbrielli, B. Koenig, P. Ganty, C. Laneve, A. Podelski, J.-F. Raskin, A. Rezine, O. Rezine, F. Rosa Velardo, A. Sangnier, J. Stuckrath, R. Traverso, G. Zavattaro, and L. Van Begin.

References

1. P. Abdulla and G. Delzanno. Constrained multiset rewriting. In *AVIS 06*, 2006.
2. P. A. Abdulla. Well (and better) quasi-ordered transition systems. *Bulletin of Symbolic Logic*, 16(4):457–515, 2010.
3. P. A. Abdulla, M. F. Atig, A. Kara, and O. Rezine. Verification of dynamic register automata. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 653–665, 2014.
4. P. A. Abdulla, M. F. Atig, and O. Rezine. Verification of directed acyclic ad hoc networks. In *FMOODS/FORTE*, pages 193–208, 2013.
5. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General decidability theorems for infinite-state systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 313–321, 1996.
6. P. A. Abdulla, G. Delzanno, and L. Van Begin. A classification of the expressive power of well-structured transition systems. *Inf. Comput.*, 209(3):248–279, 2011.

7. P. A. Abdulla, G. Delzanno, N. Ben Henda, and A. Rezine. Monotonic abstraction: on efficient verification of parameterized systems. *Int. J. Found. Comput. Sci.*, 20(5):779–801, 2009.
8. P. A. Abdulla, G. Delzanno, and A. Rezine. Approximated parameterized verification of infinite-state processes with global conditions. *Formal Methods in System Design*, 34(2):126–156, 2009.
9. P. A. Abdulla, G. Delzanno, and A. Rezine. Automatic verification of directory-based consistency protocols with graph constraints. *Int. J. Found. Comput. Sci.*, 22(4), 2011.
10. P. A. Abdulla, G. Delzanno, O. Rezine, A. Sangnier, and R. Traverso. On the verification of timed ad hoc networks. In *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, pages 256–270, 2011.
11. P. A. Abdulla, F. Haziza, and L. Holík. All for the price of few. In *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, pages 476–495, 2013.
12. P. A. Abdulla, F. Haziza, and L. Holík. Block me if you can! - context-sensitive parameterized verification. In *Static Analysis - 21st International Symposium, SAS 2014, Munich, Germany, September 11-13, 2014. Proceedings*, pages 1–17, 2014.
13. P. A. Abdulla, N. Ben Henda, G. Delzanno, and A. Rezine. Handling parameterized systems with non-atomic global conditions. In *VMCAI'08*, volume 4905 of *LNCS*, pages 22–36. Springer, 2008.
14. P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Inf. Comput.*, 130(1):71–90, 1996.
15. P. A. Abdulla and B. Jonsson. Verifying networks of timed processes (extended abstract). In *Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, TACAS '98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, pages 298–312, 1998.
16. P. A. Abdulla and B. Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theor. Comput. Sci.*, 256(1-2):145–167, 2001.
17. P. A. Abdulla and A. Nylén. Better is better than well: On efficient verification of infinite-state systems. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*, pages 132–140, 2000.
18. P.A. Abdulla, G. Delzanno, and A. Rezine. Monotonic abstraction in action. In *Theoretical Aspects of Computing - ICTAC 2008, 5th International Colloquium, Istanbul, Turkey, September 1-3, 2008. Proceedings*, pages 50–65, 2008.
19. F. Alberti, S. Ghilardi, E. Pagani, S. Ranise, and G. P. Rossi. Automated support for the design and validation of fault tolerant parameterized systems: a case study. *ECEASST*, 35, 2010.
20. B. Aminof, T. Kotek, S. Rubin, F. Spegni, and H. Veith. Parameterized model checking of rendezvous systems. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 109–124, 2014.
21. K. R. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Inf. Process. Lett.*, pages 307–309, 1986.
22. T. Ball, S. Chaki, and S. K. Rajamani. Parameterized verification of multithreaded software libraries. In *Tools and Algorithms for the Construction and Analysis of Systems, (TACAS 2001)*, pages 158–173, 2001.
23. Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci. FAST: acceleration from theory to practice. *STTT*, 10(5):401–424, 2008.
24. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, pages 101–116, 2012.
25. N. Bertrand and P. Fournier. Parameterized verification of many identical probabilistic timed processes. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, pages 501–513, 2013.
26. N. Bertrand, P. Fournier, and A. Sangnier. Playing with probabilities in reconfigurable broadcast networks. In *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 134–148, 2014.
27. N. Bertrand, P. Fournier, and A. Sangnier. Distributed local strategies in broadcast networks. In *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, pages 44–57, 2015.
28. R. Bonnet. The reachability problem for vector addition system with one zero-test. In *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, pages 145–157, 2011.
29. R. Bonnet, A. Finkel, S. Haddad, and F. Rosa-Velardo. Ordinal theory for expressiveness of well-structured transition systems. *Inf. Comput.*, 224:1–22, 2013.
30. M. Bozzano and G. Delzanno. Beyond parameterized verification. In *Tools and Algorithms for the Construction and Analysis of Systems, 8th International Conference, TACAS 2002, Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2002, Grenoble, France, April 8-12, 2002, Proceedings*, pages 221–235, 2002.
31. M. Bozzano and G. Delzanno. Automatic verification of secrecy properties for linear logic specifications of cryptographic protocols. *J. Symb. Comput.*, 38(5):1375–1415, 2004.
32. L. Bozzelli and S. Pinchinat. Verification of gap-order constraint abstractions of counter systems. *Theor. Comput. Sci.*, 523:1–36, 2014.
33. M. C. Browne, E. M. Clarke, and Orna Grumberg. Reasoning about networks with many identical finite state processes. *Inf. Comput.*, 81(1):13–31, 1989.
34. E. M. Clarke, M. Talupur, and H. Veith. Environment abstraction for parameterized verification. In *Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8-10, 2006, Proceedings*, pages 126–141, 2006.
35. V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.
36. N. David, C. Jard, D. Lime, and O. H. Roux. Discrete parameters in Petri nets. In *Application and Theory of*

Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings, pages 137–156, 2015.
37. G. Delzanno. An overview of msr(c): A clp-based framework for the symbolic verification of parameterized concurrent systems. *Electr. Notes Theor. Comput. Sci.*, 76:65–82, 2002.
38. G. Delzanno. Constraint-based verification of parameterized cache coherence protocols. *Formal Methods in System Design*, 23(3):257–301, 2003.
39. G. Delzanno. Constraint-based automatic verification of abstract models of multithreaded programs. *TPLP*, 7(1-2):67–91, 2007.
40. G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In *CSL'99*, volume 1683 of *LNCS*, pages 50–66. Springer, 1999.
41. G. Delzanno and P. Ganty. Automatic verification of time sensitive cryptographic protocols. In *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*, pages 342–356, 2004.
42. G. Delzanno, C. Di Giusto, M. Gabbrielli, C. Laneve, and G. Zavattaro. The *kappa*-lattice: Decidability boundaries for qualitative analysis in biological languages. In *CMSB*, pages 158–172, 2009.
43. G. Delzanno, J.-F. Raskin, and L. Van Begin. Towards the automated verification of multithreaded java programs. In *Tools and Algorithms for the Construction and Analysis of Systems, 8th International Conference, TACAS 2002, Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2002, Grenoble, France, April 8-12, 2002, Proceedings*, pages 173–187, 2002.
44. G. Delzanno, J.-F. Raskin, and L. Van Begin. Covering sharing trees: a compact data structure for parameterized verification. *STTT*, 5(2-3):268–297, 2004.
45. G. Delzanno and F. Rosa-Velardo. On the coverability and reachability languages of monotonic extensions of petri nets. *Theor. Comput. Sci.*, 467:12–29, 2013.
46. G. Delzanno, A. Sangnier, and R. Traverso. Parameterized verification of broadcast networks of register automata. In *RP*, pages 109–121, 2013.
47. G. Delzanno, A. Sangnier, R. Traverso, and G. Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 289–300, 2012.
48. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, pages 313–327, 2010.
49. G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 441–455, 2011.
50. G. Delzanno, A. Sangnier, and G. Zavattaro. Verification of ad hoc networks with node and communication failures. In *FORTE/FMOODS'12*, volume 7273 of *LNCS*, pages 235–250. Springer, 2012.
51. G. Delzanno and R. Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In *LATA*, pages 238–249, 2013.
52. Giorgio Delzanno and Ahmed Rezine. A lightweight regular model checking approach for parameterized systems. *STTT*, 14(2):207–222, 2012.
53. L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913.
54. G. Ding. Subgraphs and well quasi ordering. *J. of Graph Theory*, 16(5):489 – 502, 1992.
55. C. Dufourd, A. Finkel, and P. Schnoebelen. Reset nets between decidability and undecidability. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, pages 103–115, 1998.
56. E. A. Emerson and V. Kahlon. Exact and efficient verification of parameterized cache coherence protocols. In *Correct Hardware Design and Verification Methods, 12th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2003, L'Aquila, Italy, October 21-24, 2003, Proceedings*, pages 247–262, 2003.
57. E. A. Emerson and K. S. Namjoshi. On model checking for non-deterministic infinite-state systems. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 70–80, 1998.
58. E. Allen Emerson and V. Kahlon. Parameterized model checking of ring-based message passing systems. In *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, pages 325–339, 2004.
59. J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 352–359, 1999.
60. J. Esparza, P. Ganty, and R. Majumdar. Parameterized verification of asynchronous shared-memory systems. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 124–140, 2013.
61. J. Esparza, R. Ledesma-Garza, R. Majumdar, P. Meyer, and F. Niksic. An smt-based approach to coverability analysis. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, pages 603–619, 2014.
62. A. Farzan, Z. Kincaid, and A. Podelski. Proofs that count. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 151–164, 2014.
63. A. Farzan, Z. Kincaid, and A. Podelski. Proof spaces for unbounded parallelism. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 407–420, 2015.
64. A. Finkel, G. Geeraerts, J.-F. Raskin, and L. Van Begin. On the omega-language expressive power of extended Petri nets. *Electr. Notes Theor. Comput. Sci.*, 128(2):87–101, 2005.
65. A. Finkel, P. McKenzie, and C. Picaronny. A well-structured framework for analysing petri net extensions. *Inf. Comput.*, 195(1-2):1–29, 2004.
66. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

67. F. Fioravanti, A. Pettorossi, M. Proietti, and V. Senni. Improving reachability analysis of infinite state systems by specialization. *Fundam. Inform.*, 119(3-4):281–300, 2012.
68. F. Fioravanti, A. Pettorossi, M. Proietti, and V. Senni. Generalization strategies for the verification of infinite state systems. *TPLP*, 13(2):175–199, 2013.
69. P. Ganty and A. Rezine. Ordered counter-abstraction - refinable subword relations for parameterized verification. In *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, pages 396–408, 2014.
70. 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.
71. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Well-structured languages. *Acta Inf.*, 44(3-4):249–288, 2007.
72. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.
73. S. Ghilardi and S. Ranise. Backward reachability of array-based systems by SMT solving: Termination and invariant synthesis. *Logical Methods in Computer Science*, 6(4), 2010.
74. A. Gmeiner, I. Konnov, U. Schmid, H. Veith, and J. Widder. Tutorial on parameterized model checking of fault-tolerant distributed algorithms. In *Formal Methods for Executable Software Models - 14th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2014, Bertinoro, Italy, June 16-20, 2014, Advanced Lectures*, pages 122–171, 2014.
75. G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2 (7)):326–336, 1952.
76. S. Joshi and B. König. Applying the graph minor theorem to the verification of graph transformation systems. In *CAV'08*, volume 5123 of *LNCS*, pages 214–226. Springer, 2008.
77. A. Kaiser, D. Kroening, and T. Wahl. Lost in abstraction: Monotonicity in multi-threaded programs. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 141–155, 2014.
78. A. Kaiser, D. Kroening, and Thomas Wahl. A widening approach to multithreaded program verification. *ACM Trans. Program. Lang. Syst.*, 36(4):14, 2014.
79. M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.
80. R. M. Karp and R. E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, 3(2):147–195, 1969.
81. J. Kloos, R. Majumdar, F. Niksic, and R. Piskac. Incremental, inductive coverability. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 158–173, 2013.
82. B. König and V. Kozioura. Augur 2 - A new version of a tool for the analysis of graph transformation systems. *Electr. Notes Theor. Comput. Sci.*, 211:201–210, 2008.
83. D. Kroening. Automated verification of concurrent software. In *Reachability Problems - 7th International Workshop, RP 2013, Uppsala, Sweden, September 24-26, 2013 Proceedings*, pages 19–20, 2013.
84. R. Lazic, T. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundam. Inform.*, 88(3):251–274, 2008.
85. J. Leroux. The general vector addition system reachability problem by presburger inductive invariants. *Logical Methods in Computer Science*, 6(3), 2010.
86. J. Leroux. Vector addition systems reachability problem (A simpler solution). In *Turing-100 - The Alan Turing Centenary, Manchester, UK, June 22-25, 2012*, pages 214–228, 2012.
87. R. Lipton. The reachability problem requires exponential space, 1975.
88. P. Liu and T. Wahl. Infinite-state backward exploration of boolean broadcast programs. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 155–162, 2014.
89. M. Martos-Salgado and F. Rosa-Velardo. Expressiveness of dynamic networks of timed Petri nets. In *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, pages 516–527, 2014.
90. E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.
91. K. L. McMillan. Parameterized verification of the FLASH cache coherence protocol by compositional model checking. In *Correct Hardware Design and Verification Methods, 11th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2001, Livingston, Scotland, UK, September 4-7, 2001, Proceedings*, pages 179–195, 2001.
92. R. Meyer and T. Strazny. Petruchio: From dynamic networks to nets. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, pages 175–179, 2010.
93. A. Pnueli, J. Xu, and L. D. Zuck. Liveness with (0, 1, infty)-counter abstraction. In *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, pages 107–122, 2002.
94. G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *ACM Trans. Program. Lang. Syst.*, 22(2):416–430, 2000.
95. K. Reinhardt. Reachability in Petri nets with inhibitor arcs. *Electr. Notes Theor. Comput. Sci.*, 223:239–264, 2008.
96. P. Z. Revesz. A closed-form evaluation for datalog queries with integer (gap)-order constraints. *Theor. Comput. Sci.*, 116(1):117–149, 1993.
97. F. Rosa-Velardo and D. de Frutos-Escrig. Decidability results for restricted models of Petri nets with name creation and replication. In *Applications and Theory of Petri Nets (PETRI NETS 2009)*, pages 63–82, 2009.
98. F. Rosa-Velardo and D. de Frutos-Escrig. Decidability and complexity of Petri nets with unordered data. *Theor. Comput. Sci.*, 412(34):4439–4451, 2011.
99. M. Saksena, O. Wibling, and B. Jonsson. Graph grammar modeling and verification of ad hoc routing protocols. In *TACAS*, pages 18–32, 2008.
100. P. Schnoebelen. Revisiting ackermann-hardness for lossy counter machines and reset Petri nets. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 616–628, 2010.
101. P. Schnoebelen. Revisiting ackermann-hardness for lossy counter machines and reset Petri nets. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, pages 616–628, 2010.

102. A. Singh, C. R. Ramakrishnan, and S. A. Smolka. Query-based model checking of ad hoc network protocols. In *CONCUR*, pages 603–619, 2009.
103. A. Singh, C. R. Ramakrishnan, and S. A. Smolka. A process calculus for mobile ad hoc networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.
104. J. Stückrath. Uncover: Using coverability analysis for verifying graph transformation systems. In *Graph Transformation - 8th International Conference, ICGT 2015, Held as Part of STAF 2015, L'Aquila, Italy, July 21-23, 2015. Proceedings*, pages 266–274, 2015.
105. `https://github.com/pierreganty/mist`.
106. `http://users.mat.unimi.it/users/ghilardi/mcmt/`.
107. `http://www.ahmedrezine.com/tools/`.
108. `http://www.ccs.neu.edu/home/wahl/Research/boom-and-cutoffs.html`.
109. `http://www.disi.unige.it/person/DelzannoG/MSR/`.
110. `http://www.it.uu.se/research/docs/fm/apv/tools/pfs/`.
111. `http://www.it.uu.se/research/docs/fm/apv/tools/undip/`.
112. `http://www.liafa.jussieu.fr/~sighirea/trex/`.
113. `http://www.lsv.ens-cachan.fr/Software/fast/`.
114. `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.
115. `http://www.ti.inf.uni-due.de/de/research/tools/uncover/`.
116. `http://www.ulb.ac.be/di/verif/ggeeraer/eec/`.