Efficient Local and Global Model Checking

B.–H. Schlingloff, Technische Universität München

schlingl@informatik.tu-muenchen.de

In order to apply formal verification methods to industrial-size product developments there are two prerequisites: First, one needs convenient, intuitive description and specification languages for the system and properties to be verified. Second, there should be efficient algorithms with reasonable average-case complexity for the analysis of the formal languages under consideration.

Elementary Petri nets provide a natural way to model the control structure of concurrent and distributed systems. Since the state spaces of elementary nets are finite, most properties of these state spaces are decidable. In contrast to other finite-state representations of relational structures, Petri nets distinguish between nondeterminism caused by the abstraction from data, and nondeterminism caused by different scheduling strategies of the parallel processes. As we show, this fact can be exploited to improve the evaluation of certain properties.

Formally, an EPN is a tuple $N \triangleq (P, T, \mu_0, pre, post)$, where P and T are nonempty finite sets of *places* and *transitions*, respectively, $\mu_0 \in 2^P$ is the *initial marking* of the net, and $pre, post : T \mapsto 2^P$ are functions determinining the *pre-* and *postset* of a transition, respectively. A *marking* $\mu \in 2^P$ is any subset of P, and for any $t \in T$ we define the *firing relation* $\xrightarrow{t}: 2^P \mapsto 2^P$ as follows: $\mu_1 \xrightarrow{t} \mu_2$ if $pre(t) \subseteq \mu_1$, $post(t) \cap \mu_1 = \emptyset$, and $\mu_2 = (\mu_1 \setminus pre(t)) \cup post(t)$. The *reachable state space* of an EPN is the smallest set of markings containing μ_0 which is closed under all firing relations. An *execution* is a maximal path through the reachable state space. The EPN is *deadlock-free* if every execution is infinite.

Temporal Logic was invented by philosophers to formally duplicate natural language sentences about events in time. Several variants and extensions for the specification of reactive systems have been developed, most notably being the distinction between branching time logics (interpreted on reachable state spaces) and linear time logics (interpreted on executions). In this note, we use dynamic Peirce algebras to represent certain properties of elementary nets, since they can be interpreted both on branching and linear structures.

A DPA is a tuple $K \triangleq (A, B, :, ?)$, where $A \triangleq (T, \sqcup, \overline{\ }, \perp\!\!\!\perp, \cdot, \breve{\ }, \mathbb{I}, {}^*)$ is a relation algebra with constant elements, join, complement, zero, composition, converse, identity and Kleene star; $B \triangleq (P, \vee, \neg, \perp)$ is a boolean algebra; ":" is a Peirce product (modal diamond) $A \times B \mapsto B$, and "?" is a cylindrification mapping (test) $B \mapsto A$. The *concrete interpretation* $\mathcal{I}_N(\varphi)$ of a DPA term φ in an EPN N assigns the relation $\xrightarrow{t}$ to every relation constant $t \in T$, and the set $\{\mu \mid p \in \mu\}$ of markings to every boolean constant $p \in P$. All other operators are assigned their usual relation algebraic or set theoretic meanings, e.g., τ^* becomes the reflexive transitive closure of the relation τ. For $\varphi \in B$, we have $\mathcal{I}_N(\varphi?) \triangleq \{(\mu, \mu) \mid \mu \in \mathcal{I}_N(\varphi)\}$. Finally, for $\varphi \in B$ and $\tau \in A$, we have $\mathcal{I}_N(\tau : \varphi) \triangleq \{\mu \mid \exists \nu \in \mathcal{I}_N(\varphi),\ (\mu, \nu) \in \mathcal{I}_N(\tau)$.

Model checking is the process of determining whether the execution of the system modelled by an EPN satisfies the properties descriped by a DPA term. For example, deadlock freedom can be described by the term $\neg(T^* : \neg(T : \neg\perp))$. There are two basic approaches to the model checking problem: In *global* model checking we try to build the set of markings satisfying a given term by recursion on the structure of the term. Since the number of reachable markings can be exponential in the number of places, often it is not possible to

use an explicit enumeration. We use a *symbolic* representation as binary decision diagrams (BDDs, [Bry92]) to maintain large sets and relations.

The function *eval* assign the BDD of a relation to all elements of A, and the BDD of a set of markings to all elements of B which occur in the given term. The base cases are $eval(t) \triangleq BDD(\xrightarrow{t})$ and $eval(p) \triangleq BDD(\{\mu \mid p \in \mu\})$; the size of these BDDs is linear in the number of places and constant, respectively. All boolean and relation algebraic operations can be performed directly using BDDs; complement is particularly easy since it is done in constant time. Building the join of two BDDs is a potentially exponential step. Relation composition is an existential quantification on an intermediate marking, and the Kleene star is calculated as the least fixpoint of a relation composition. The BDD for $p?$ is twice as big as the BDD for p, and the BDD for $eval(t : p)$ is the preset of $BDD(p)$ under $BDD(t)$, which can be calculated efficiently using special techniques ([CGL93]).

Local model checking corresponds to the linear-time approach: we try to systematically construct an execution (dis)satisfying a given formula. To do so, we build the product of the *tableau* of a formula with the unfolding of the net. For example, with marking μ, the term $(\tau : \varphi)$ is satisfiable iff there exists a marking ν such that $\mu \xrightarrow{\tau} \nu$ and φ is satisfiable with marking ν. Systematic depth-first-search will either fall into a loop or produce a counterexample. A loop is satisfiable if it does not contain unsatisfied eventualities, i.e., formulas $(\tau : \varphi)$ for which no formula φ occurs in the loop.

Local model checking can be improved for certain classes of formulas by building only *part* of the reachable state space ([Val91]). Two executions are stuttering equivalent w.r.t. a subset of all places and transitions, if any formula built solely from this subset has the same truth value on both executions. Transition t_1 is *independent* from t_2 w.r.t. marking μ and formula ϕ, if for any execution starting with t_2 there exists a stuttering equivalent execution starting with t_1. In this case we do not need to consider the firing of t_2 from marking μ for evaluating the given formula. Depending on the type of formula, there are syntactical criteria to determine whether a transition could be independent from another. A detailed treatment for real-time temporal logic can be found in [YS95]

We have combined the partial order technique and the symbolic method for the analysis of deadlock-freedom. For the symbolic search this means we augmented the algorithm by a component reducing the number of markings which have to be considered in a single iteration step. For the partial order search this means a generalization of the algorithm such that in every step a set of markings is considered and the iteration step is performed breadth-first instead of depth-first.

First experiments show that the combination of partial and symbolic method is especially useful for massively parallel systems, in which several small components work more or less independently. This suggests that the combined method can be applied successfully to object-oriented designs. Currently we are extending our algorithm to handle the full expressiveness of stuttering invariant temporal logics.

References

[Bry92] R. Bryant: Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams; Rep. CMU-CS-92-160;
`ftp://reports.adm.cs.cmu.edu/usr/anon/1992/CMU-CS-92-160.ps`

[CGL93] E. Clarke, O. Grumberg, D. Long: Verification Tools for Finite-State Concurrent Systems; LNCS 803 (1993)

[YS95] T. Yoneda, B.-H. Schlingloff: Efficient Verification of Parallel Real-Time Systems; LNCS 697 (1993); revised:
`http://www1.informatik.tu-muenchen.de/publikationen/tempspez`

[Val91] A. Valmari: A stubborn attack on state explosion; Proc CAV 90.