Proving Properties of Elementary Net Systems with a Special-Purpose Theorem Prover

Heikki Tuominen
Nokia Telecommunications
P.O.Box 33, SF-02601 ESPOO, Finland
e-mail:htuomine@tele.nokia.fi

Abstract

Elementary net systems, basic net-theoretic models of distributed systems, as well as their properties are represented by formulas of deterministic propositional dynamic logic. A semantic tableaux based theorem prover is used to verify the properties, i.e. to find out whether they are entailed by the representations of the systems. The theorem prover implements a simplified tableaux decision procedure, the simplifications are due to the special nature of the set of formulas representing an elementary net system.

1 Introduction

Elementary net systems — EN systems in the sequel — are basic net-theoretic models of distributed systems [5]. In addition to an intuitive graphical representation EN systems offer various mathematical analysis methods. These make it possible for the designer to verify his/her design already prior to the implementation. However, the analysis of EN systems is quite a diverse field: there are separate algorithms for verifying properties of EN systems but some uniformity is missing. One way to unify the analysis is to use a formal language to represent the properties and a general enough algorithm to verify those properties.

EN systems as well as their properties can be represented by formulas of deterministic propositional dynamic logic (DPDL) in a natural way. The most obvious way of representing an EN system is to axiomatize the set of dynamic logic formulas true in the initial state of its state graph [6]. The representation as a set of formulas implies that a single algorithm, a decision procedure for the logic, can be used to prove different properties of the EN system. Speaking in terms of logic the verification problem is thus to find out whether a formula representing a property is entailed by a set of formulas representing an EN system.

Having a set of DPDL formulas Γ representing an EN system in the described sense and a formula ϕ representing a property of the system the tableaux-based decision procedure for DPDL [1] can be used to find out whether $\Gamma \models \phi$ or not. The procedure is applied as a systematic attempt to construct a model M with a state s such that $M, s \models \Gamma \cup \{\neg\phi\}$ and if the construction fails the entailment holds. However, knowing that the set of premises Γ represents an EN system allows significant simplifications in the decision procedure.

There are two main sources for the simplification of the decision procedure. Firstly, most of the formulas representing an EN system have a global nature, i.e. they express restrictions which should be true in any state of the system.

Thus a decision procedure for the global entailment relation is more suitable than the standard local one. Instead of trying to construct a model with a state in which the premises are true and the conclusion false a procedure for the global entailment relation tries to construct a model such that the premises are true in all of its states and the conclusion false in same of its states. The original decision procedure can quite easily be modified in this respect by an additional parameter for the global premises. Secondly, the models of a set of formulas representing an EN system possess a "strong" collapsed model property, i.e. any two states verifying exactly the same atomic formulas in such a model can be identified without affecting the truth values of relevant formulas. This is due to the fact that the models correspond to the state graph of the EN system in which the successors of a state are completely determined by the atomic formulas true in that state.

Even with the simplified decision procedure it is in most non-trivial cases impossible in practice to prove $\Gamma \models \phi$ when the system represented by Γ really has the property represented by ϕ. However, it is often possible to prove that the system does not have the opposite property represented by $\neg\phi$, i.e. $\Gamma \not\models \neg\phi$ [7]. These are equivalent because of the completeness of the logical theory induced by Γ. The indirect method has one drawback: in order to prove that an EN system does not have a property the whole case graph has to be constructed. This construction can be avoided only by using some other representation of the EN system, a representation which axiomatizes only a subset of the properties. One such approach presented in this paper is to axiomatize the set of safety properties. The simplifications of the decision procedure apply also for this representation although for partly different reasons.

Another method to verify properties of EN systems would be the model checking paradigm: generate the state graph of the system, interpret it as a data base and use a temporal (or dynamic) logic as a query language [2]. The present theorem proving method goes one step further in utilizing logic and represents also the system as a set of formulas. This makes the method more general and flexible, the same algorithm can be used in connection with different representations of EN systems. Unfortunately this generality also means a certain loss in the efficiency.

The rest of the paper is organized as follows. Section 2 is an informal review of EN systems. Two different ways to represent them as formulas of dynamic logic are discussed. The simplified decision procedure for DPDL is described in Section 3 and its experimental implementation is used to verify properties of EN systems in Section 4. Conclusions are drawn in Section 5.

2 Elementary net systems and their representation in dynamic logic

Elementary net systems belong to the family of Petri nets [5]. Graphically EN systems are represented as directed graphs with two kinds of nodes, circles and boxes, e.g. Figure 1. The circles, called conditions, represent the local state elements and the boxes, called events, represent the local actions. In the graphical representation the state of the system is indicated by tokens in some of the conditions, b_1 and b_2 in the example. The conditions having a token in a state, called a case, are interpreted to be true in that state, the others false. $\{b_1, b_2\}$ forms the initial

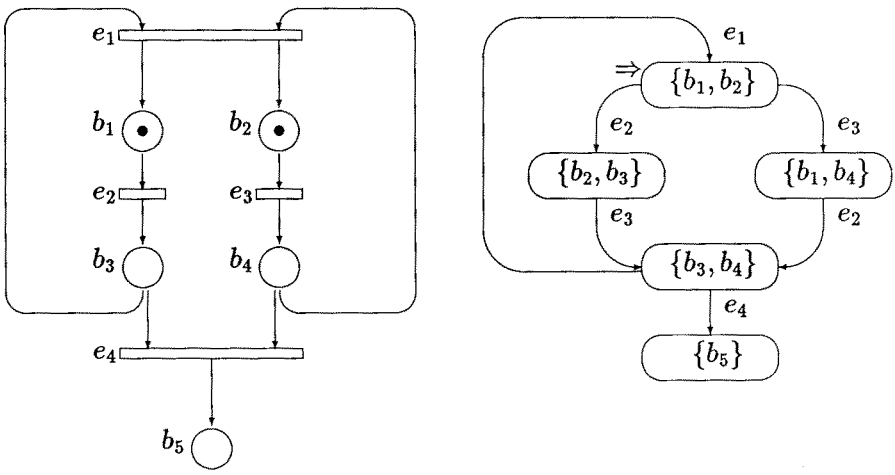

Figure 1: An EN system and its sequential case graph.

case of the EN system in Figure 1. The state of an EN system is changed by occurrences of the events. An event can occur if all of its pre-conditions (conditions from which there is an arc to the event) are true and all of its post-conditions false. The event occurrence causes the pre-conditions to become false and the post-conditions to become true.

The sequential case graph of an EN system represents the forward state graph of the system, it is an initialized labelled graph [4]. The nodes are labelled with sets of conditions, the intuition being that those conditions are true in the case which that node represents. The initial node is labelled with the set of conditions true in the initial case of the system. The edges of the sequential case graph are labelled with events of the system. Intuitively an edge represents the event whose occurrence takes the system from a case to another. In this paper we restrict ourselves to finite EN systems which always have also finite sequential case graphs.

It is obvious that sequential case graphs are exactly like models of propositional dynamic logic, a branch of modal logic originally developed for reasoning about computer programs [3]. The events and conditions correspond directly to atomic programs and formulas, respectively. While sequential case graphs are deterministic structures they are even models of DPDL, a deterministic propositional dynamic logic [1].

In DPDL programs are regular expressions over the set of atomic programs and test programs, i.e. they are built using concatenation (;), choice ($\cup$) and finite repetition ($*$) from the atomic programs and expressions of the form $\phi?$ where ϕ is a formula. Formulas are built from atomic formulas using negation ($\neg$) and prefixing with $\langle\alpha\rangle$, where α is any program. The intended reading of the formula $\langle\alpha\rangle\phi$ is "it is possible to execute α and reach a state in which ϕ is true." The dual of $\langle\alpha\rangle$ and the additional truth-functional operators are defined as abbreviations in the usual way: $[\alpha]\phi$ stands for $\neg\langle\alpha\rangle\neg\phi$, $\phi \wedge \psi$ stands for $\langle\phi?\rangle\psi$, $\phi \vee \psi$ stands for $\neg(\neg\phi \wedge \neg\psi)$, $\phi \rightarrow \psi$ stands for $\neg\phi \vee \psi$, and $\phi \leftrightarrow \psi$ stands for

$(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$.

EN systems can be represented as sets of DPDL formulas in several ways. Perhaps the most obvious way is to axiomatize the theory formed by formulas true in the initial case of the sequential case graph. The set of non-logical axioms doing this for an EN system $\mathcal{N}$ is in the sequel denoted by $\Gamma^a_{\mathcal{N}}$. For the EN system in Figure 1 the set is the following.

$(b_1 \wedge b_2 \wedge \neg b_3 \wedge \neg b_4 \wedge \neg b_5)$

$$
\begin{array}{ll}
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((\neg b_1 \wedge \neg b_2 \wedge b_3 \wedge b_4) & \rightarrow \langle e_1 \rangle (b_1 \wedge b_2 \wedge \neg b_3 \wedge \neg b_4)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_1 \wedge \neg b_3) & \rightarrow \langle e_2 \rangle (\neg b_1 \wedge b_3)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_2 \wedge \neg b_4) & \rightarrow \langle e_3 \rangle (\neg b_2 \wedge b_4)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_3 \wedge b_4 \wedge \neg b_5) & \rightarrow \langle e_4 \rangle (\neg b_3 \wedge \neg b_4 \wedge b_5))
\end{array}
$$

$$
\begin{array}{ll}
[(e_1 \cup e_2 \cup e_3 \cup e_4)*](\neg(\neg b_1 \wedge \neg b_2 \wedge b_3 \wedge b_4) & \rightarrow [e_1]\bot) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*](\neg(b_1 \wedge \neg b_3) & \rightarrow [e_2]\bot) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*](\neg(b_2 \wedge \neg b_4) & \rightarrow [e_3]\bot) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*](\neg(b_3 \wedge b_4 \wedge \neg b_5) & \rightarrow [e_4]\bot)
\end{array}
$$

$$
\begin{array}{lll}
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_1 \rightarrow [e_3 \cup e_4]b_1) & \wedge (\neg b_1 \rightarrow [e_3 \cup e_4]\neg b_1)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_2 \rightarrow [e_2 \cup e_4]b_2) & \wedge (\neg b_2 \rightarrow [e_2 \cup e_4]\neg b_2)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_3 \rightarrow [e_3]b_3) & \wedge (\neg b_3 \rightarrow [e_3]\neg b_3)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_4 \rightarrow [e_2]b_4) & \wedge (\neg b_4 \rightarrow [e_2]\neg b_4)) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*]((b_5 \rightarrow [e_1 \cup e_2 \cup e_3]b_5) & \wedge (\neg b_5 \rightarrow [e_1 \cup e_2 \cup e_3]\neg b_5))
\end{array}
$$

The axioms are grouped into four sets. The first group is formed by a formula which defines the initial case. The second group includes for each event of the system a formula which describes the occurrence of that event. The third group includes for each event a formula which defines in which cases that event cannot occur. The last group includes the so called frame axioms, one for each condition. The described set of axioms represents the EN system in the sense that a formula (in the relevant language) is entailed by the set in DPDL if and only if the formula is true in the initial case of the sequential case graph of the system [6].

Another way to represent an EN system as a set of DPDL formulas is to axiomatize only the safety properties, i.e. formulas which state that nothing bad will happen. That set of non-logical axioms $\Gamma^b_{\mathcal{N}}$ differs from $\Gamma^a_{\mathcal{N}}$ only in the second group. Instead of stating that an enabled event must occur, the new axioms state that whenever the event occurs the resulting case has the pre-conditions false and the post-conditions true. The second group of axioms in $\Gamma^b_{\mathcal{N}}$ for the EN system in Figure 1 is the following.

$$
\begin{array}{l}
[(e_1 \cup e_2 \cup e_3 \cup e_4)*][e_1](b_1 \wedge b_2 \wedge \neg b_3 \wedge \neg b_4) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*][e_2](\neg b_1 \wedge b_3) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*][e_3](\neg b_2 \wedge b_4) \\
[(e_1 \cup e_2 \cup e_3 \cup e_4)*][e_4](\neg b_3 \wedge \neg b_4 \wedge b_5)
\end{array}
$$

This representation is closer to the original idea of EN systems (and Petri nets in general): they only define causal dependencies between occurrences of events without claiming that some event will occur.

The latter representation is entailed by the former one, i.e. $\Gamma^a_{\mathcal{N}} \models \Gamma^b_{\mathcal{N}}$. Thus if a formula ϕ is entailed by the latter representation it also is entailed by the former one. For formulas representing safety properties also the converse holds as the following proposition states.

Proposition 2.1 *If the Fischer-Ladner closure* [1] *of ϕ includes no formula of the form $\langle\alpha\rangle\psi$ (where α is an atomic program) which appears in ϕ in the scope of an even number of negation symbols and $\Gamma^a_{\mathcal{N}} \models \phi$ then $\Gamma^b_{\mathcal{N}} \models \phi$.*

The proposition together with the earlier mentioned entailment implies that regarding safety properties of the EN system the two representations are equivalent. This relation makes it possible to prove that a system does not have a certain safety property without generating the whole case graph. Examples of using both representations in reasoning about EN systems are given in Section 4.

3 The simplified tableaux procedure

The decision procedure which we use to prove properties of EN systems is based on the semantic tableaux algorithm for DPDL [1]. The original algorithm works in two phases: given a satisfiable formula (or a set of formulas) it first generates a structure called a partial D model and then unwinds the structure into a model. The partial D model consists of a relevant part of a filtration of the canonical model through the Fischer-Ladner closure of the original formula and it is otherwise like a model but the accessibility relations corresponding to atomic programs are not necessarily functional.

There are at least two simplifications possible in the decision procedure when it is applied to a representation of an EN system, $\Gamma^a_{\mathcal{N}}$ or $\Gamma^b_{\mathcal{N}}$. The first one is based on the syntactic form of the representations, the second one on properties of their models.

The syntactically motivated simplification is to change the algorithm to operate on the global entailment relation. The change simplifies the proof of properties because most of the formulas representing an EN system are global by nature, i.e. they have the "henceforth" prefix, $[(\cup e_i)*]$. Instead of using the local entailment relation $\Gamma_{\mathcal{N}} \models \phi$, where $\Gamma_{\mathcal{N}}$ (being $\Gamma^a_{\mathcal{N}}$ or $\Gamma^b_{\mathcal{N}}$) represents the system and ϕ the property, it is thus more reasonable to use the global relation $\Gamma'_{\mathcal{N}} \models_G (\psi \rightarrow \phi)$, where ψ is the formula in $\Gamma_{\mathcal{N}}$ which defines the initial case and $\Gamma'_{\mathcal{N}}$ consists of the rest of formulas in $\Gamma_{\mathcal{N}}$ leaving out the "henceforth" prefixes. The reading of $\Gamma'_{\mathcal{N}} \models_G (\psi \rightarrow \phi)$ is: if $\Gamma'_{\mathcal{N}}$ is true in all states of a model then $(\psi \rightarrow \phi)$ is also true in any state of such a model. Clearly $\Gamma_{\mathcal{N}} \models \phi$ iff $\Gamma'_{\mathcal{N}} \models_G (\psi \rightarrow \phi)$. In the algorithm this change appears as an additional parameter for the set $\Gamma'_{\mathcal{N}}$ of global premises. The algorithm is run on the set $\Gamma'_{\mathcal{N}} \cup \{\psi, \neg\phi\}$ and every time a new node of the partial D model is generated by the γ-rule the members of $\Gamma'_{\mathcal{N}}$ are added to the label of that new node.

The semantically motivated simplification is due to the following properties of the models of $\Gamma^a_{\mathcal{N}}$ and $\Gamma^b_{\mathcal{N}}$.

[1] The Fischer-Ladner closure of a formula ϕ is the least set of formulas Σ which includes ϕ and satisfies the following conditions: if $\neg\psi \in \Sigma$ then $\psi \in \Sigma$, if $\langle\alpha\rangle\psi \in \Sigma$ then $\psi \in \Sigma$, if $\langle\alpha;\beta\rangle\psi \in \Sigma$ then $\langle\alpha\rangle\langle\beta\rangle\psi \in \Sigma$, if $\langle\alpha \cup \beta\rangle\psi \in \Sigma$ then $\{\langle\alpha\rangle\psi, \langle\beta\rangle\psi\} \subseteq \Sigma$, if $\langle\alpha*\rangle\psi \in \Sigma$ then $\langle\alpha\rangle\langle\alpha*\rangle\psi \in \Sigma$, and if $\langle\psi?\rangle\chi \in \Sigma$ then $\{\psi, \chi\} \subseteq \Sigma$ [1].

size of the system	size of the case graph	execution time (s)
9	5	1
13	10	4
17	17	10
21	26	21
25	37	69
29	50	126
33	65	227
37	82	376

Table 1: Runtime statistics for case graph generation.

- In any model M generated by a state s such that $M, s \models \Gamma^a_{\mathcal{N}}$ the successors of any state are completely determined by the atomic formulas true in that state.
- The representation $\Gamma^b_{\mathcal{N}}$ has the following "finite tree model property": if ϕ represents a safety property and $\Gamma^b_{\mathcal{N}} \cup \{\neg\phi\}$ is satisfiable then there exists a finite tree model M generated by a state s such that $M, s \models \Gamma^b_{\mathcal{N}} \cup \{\neg\phi\}$.

These imply that the tableaux algorithm can be modified to construct directly a model. This is done by using a generate and test methodology in the sense that every time the algorithm meets a disjunctive (β) formula the construction based on the first alternative (β_1) is tried first. If that choice does not lead to a model in which all the eventuality formulas get fulfilled the second alternative (β_2) is tried.

4 Experimental results

An experimental theorem prover based on the described simplified decision procedure has been implemented in Prolog. This section contains runtime statistics for different uses of the prover run on a VAX 8650 mainframe computer.

4.1 Logic-based case graph generation

The representation $\Gamma^a_{\mathcal{N}}$ of an EN system $\mathcal{N}$ which axiomatizes the set of formulas true in the initial case of the sequential case graph can be used for logic-based case graph generation. The model generated by the tableaux procedure corresponds to the sequential case graph of the represented EN system. Using the notation defined in Section 3 a model can be generated by testing whether $\Gamma^{a\prime}_{\mathcal{N}} \models_G (\psi \rightarrow \bot)$. Table 1 contains runtime statistics for this test considering a set of EN systems. The smallest system is the one in Figure 1, in the second a condition-event pair is added between e_2 and b_3 as well as e_3 and b_4, in the third system again two additional pairs are added etc. and the enumeration of the elements is changed accordingly. The number of conditions and events is used as a measure of the size of the system.

formula	size of the falsifying model	execution time (s)
$[e_2]\bot$	2	9
$[e_2;e_3]\bot$	3	9
$[e_2;e_3;e_4]\bot$	4	11
$[e_2;e_3;e_4;e_5]\bot$	5	12
$[e_2;e_3;e_4;e_5;e_6]\bot$	6	13
$[e_2;e_3;e_4;e_5;e_6;e_7]\bot$	7	15
$[e_2;e_3;e_4;e_5;e_6;e_7;e_8]\bot$	8	16
$[e_2;e_3;e_4;e_5;e_6;e_7;e_8;e_9]\bot$	9	17
$[e_2;e_3;e_4;e_5;e_6;e_7;e_8;e_9;e_{10}]\bot$	10	19

Table 2: Runtime statistics for event occurrence sequences.

formula	size of the falsifying model	execution time (s)
$[(\bigcup_{i=1...18} e_i)*]\neg b_1$	1	8
$[(\bigcup_{i=1...18} e_i)*]\neg b_5$	4	16
$[(\bigcup_{i=1...18} e_i)*]\neg b_9$	8	20
$[(\bigcup_{i=1...18} e_i)*]\neg b_{13}$	12	57
$[(\bigcup_{i=1...18} e_i)*]\neg b_{17}$	16	72

Table 3: Runtime statistics for reachable cases.

4.2 Proving properties

Even using the simplified tableaux procedure it is impossible to prove — within reasonable time — that ${\Gamma^a_{\mathcal{N}}}' \models_G (\psi \to \phi)$ for most non-trivial properties ϕ if the system really has the property ϕ. However, it is often possible to prove that the system does not have the opposite property represented by $\neg\phi$ [7]. These are equivalent because of the completeness of the theory induced by the representation, i.e. for each formula ϕ (in the relevant language) either ${\Gamma^a_{\mathcal{N}}}' \models_G (\psi \to \phi)$ or ${\Gamma^a_{\mathcal{N}}}' \models_G (\psi \to \neg\phi)$. Thus ${\Gamma^a_{\mathcal{N}}}' \models_G (\psi \to \phi)$ iff ${\Gamma^a_{\mathcal{N}}}' \not\models_G (\psi \to \neg\phi)$. The main drawback of this indirect method is that in order to prove that an EN system $\mathcal{N}$ does not have a property using the representation ${\Gamma^a_{\mathcal{N}}}'$ the whole case graph has to be constructed. This means that the execution time needed to verify a property is always greater than the time needed to construct the case graph.

However, as explained in Section 2 for safety properties the situation is better because the representation $\Gamma^b_{\mathcal{N}}$ can be used. Table 2 contains runtime statistics for proofs of possible event occurrence sequences in the largest system in Table 1, the one with 37 elements. The representation ${\Gamma^b_{\mathcal{N}}}'$ of that EN system is used. Similarly Table 3 contains statistics for proofs of the accessibility of cases with a given condition true.

5 Conclusions

It was shown to be possible — at least in principle — to prove properties of elementary net systems with an automated theorem prover for dynamic logic. Compared with the model checking approach the performance is nevertheless quite poor.

Acknowledgements

The financial support for this work was provided by the Academy of Finland.

References

[1] Ben-Ari, M., Halpern, J., and Pnueli, A. Deterministic Propositional Dynamic Logic: Finite Models, Complexity, and Completeness. *Journal of computer and system sciences 25*, (1982), 402–417.

[2] Clarke, E., Browne, M., Emerson, E., and Sistla, A. Using Temporal Logic for Automatic Verification of Finite State Systems. In *Logics and Models of Concurrent Systems*, K. Apt, Ed., Springer-Verlag, Berlin, 1985, pp. 3–26.

[3] Harel, D. Dynamic logic. In *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, D. Gabbay and F. Guenthner, Eds., D. Reidel Publishing Company, Dordrecht, 1984, pp. 497–604.

[4] Rozenberg, G. Behaviour of Elementary Net Systems. In *Petri Nets: Central Models and Their Properties (Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.–19. September 1986)*, W. Brauer, W. Reisig, and G. Rozenberg, Eds., Springer-Verlag, Berlin, 1987, pp. 60–94.

[5] Thiagarajan, P. Elementary Net Systems. In *Petri Nets: Central Models and Their Properties (Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, 8.–19. September 1986)*, W. Brauer, W. Reisig, and G. Rozenberg, Eds., Springer-Verlag, Berlin, 1987, pp. 26–59.

[6] Tuominen, H. Elementary Net Systems and Dynamic Logic. In *IX European Workshop on Application and Theory of Petri Nets* (Venice, Italy, June 22–24). 1988, pp. 24–37.

[7] Tuominen, H. Proving Properties of Elementary Net Systems with an Automated Theorem Prover. In *COLOG-88, Papers presented at the International Conference in Computer Logic, Part I* (Tallinn, Dec. 12–16), P. Lorentz, G. Mints, and E. Tyugu, Eds., Institute of Cybernetics of the Academy of Sciences of the Estonian SSR, 1988, pp. 206–220.