

Modal vs. Propositional Reasoning for Model Checking with Description Logic

Shoham Ben-David Richard Trefler Grant Weddell

David R. Cheriton School of Computer Science
University of Waterloo

1 Introduction

Model checking ([7, 13], c.f.[6]) is a technique for verifying finite-state concurrent systems that has proven effective in the verification of industrial hardware and software programs. In model checking, a model M , given as a set of state variables V and their next-state relations, is verified against a temporal logic formula φ . In this work we consider only safety formulas of the form $AG(b)$, with b being a Boolean expression over the state variables of the model, meaning that b is an invariant of M .

The main challenge in model checking is known as the *state space explosion* problem, where the number of states in the model grows exponentially in the number of program variables. To cope with this problem, model checking is done *symbolically*, by representing the system under verification as sets of states and transitions, and by using Boolean functions to manipulate those sets. Two main symbolic methods are used to perform model checking. The first, implemented in *SMV* [10], is based on Binary Decision Diagrams (BDDs) [5]. The second is known as Bounded Model Checking (*BMC*) [4]. Using this method, the model under verification and its specification are unfolded to depth k (for a given bound k), and translated into a propositional CNF formula. This results in an encoding of the model checking problem that is essentially k times the size of the textual description of M . A SAT solver is then applied to the formula to find a satisfying assignment. Such an assignment, if found, demonstrates an error in the model.

We investigate the possibility of using a Description Logic reasoner for bounded model checking. Recent work [3] showed how to embed BMC problems as concept consistency problems in the DL dialect *ALCT*. The encoding as a terminology resulted in a natural *symbolic* representation of the sets of states and transitions that is significantly smaller than the one obtained by translating a model into a CNF formula. This translation works as follows.

Let M be a model defined by a set V of Boolean state variables and their next-state transitions R . We represent each variable $v_i \in V$ as a concept V_i , and the transition relation as a single role R . We then introduce concept inclusions of the type

$$C_0 \sqsubseteq \forall R.C_1$$

stating that if the current state satisfies the condition represented by C_0 then all the next-states that can be reached in one step by R must satisfy the condition C_1 . Note that interpretations for this set of concept inclusions correspond to sub-models of M .

Let the concept S_0 represent the set of initial states of M , and let S_1 be a new concept. If the concept inclusion $S_1 \sqsubseteq \exists R^-.S_0$ holds in the interpretation, then the set S_1 is a subset of all the states that can be reached from S_0 by going one step forward

using the relation R . Similarly, we denote by S_i a subset of the states that can be reached after i steps, and introduce the inclusions

$$S_i \sqsubseteq \exists R^- . S_{i-1}$$

for $0 < i \leq k$. Let $\varphi = AG(b)$ be the specification to be verified, and let B be the concept representing b (composed of a Boolean combination of the concepts V_i representing the state variables). Model checking is then carried out by asking the query: “does there exist an interpretation for the above set of concept inclusions such that $\neg B \sqcap S_i$ is not empty for some S_i ?”. A positive answer from the DL reasoner indicates an error in M .

Experimental results comparing this method to SAT-based model checking showed that SAT solving outperformed DL reasoning, especially as the bound k increased [3], despite the significantly smaller DL encoding of the model checking problem — k times the size of M in the SAT case vs. k plus the size of M in the DL case.

In this paper we report on an experiment aimed at determining whether it is the *modal* reasoning (that involved taking backward steps through the role R) that caused the problem, or the propositional reasoning that is more efficient in the SAT solvers. For this we produced a series of translations of the model into a terminology. Instead of one set of concepts corresponding to the variables of the given model, we introduce l copies to represent states of increasing distance from the initial state. The number of *modal* traversals through R is then reduced by a factor of l while increasing the size of the DL encoding by a factor of l . We expected the reduced modality translations would outperform the original. It was surprising to us that the results suggested the opposite.

The rest of the paper is organized as follows. In the next section we give definitions and present bounded model checking using DL [3]. Section 3 is the main section of the paper where the new translation is presented. An evaluation and discussion follow in Section 4.

2 Background and Definitions

Definition 1 (Description Logic \mathcal{ALCI}) Let NC and NR be sets of atomic concepts $\{A_1, A_2, \dots\}$ and atomic roles $\{R_1, R_2, \dots\}$ respectively. The set of concepts C of the description logic \mathcal{ALCI} is the smallest set including NC that satisfies the following.

- If $C_1, C_2 \in C$ then so are $\neg C_1$ and $C_1 \sqcap C_2$
- If $C \in C$ and $R \in R$ then so are $\exists R.C$ and $\exists R^- .C$

Additional concepts are defined as syntactic sugaring of those above:

- $\top = A \sqcup \neg A$ for some A
- $\forall R.C = \neg \exists R. \neg C$
- $C_1 \sqcup C_2 = \neg(\neg C_1 \sqcap \neg C_2)$

A *General Concept Inclusion Axiom* is an expression of the form $C_1 \sqsubseteq C_2$. A *TBox* \mathcal{T} consists of a finite set of inclusion dependencies.

The *semantics* of expressions is defined with respect to a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function mapping every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the following conditions are satisfied.

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \sqcap C_2^{\mathcal{I}}$
- $\exists R.C = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
- $\exists R^- .C = \{y \in \Delta^{\mathcal{I}} \mid \exists x \in \Delta^{\mathcal{I}} \text{ s.t. } (x, y) \in R^{\mathcal{I}} \wedge x \in C^{\mathcal{I}}\}$

A structure *satisfies an inclusion dependency* $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. The *consistency problem for \mathcal{ALCC}* asks if $\mathcal{T} \models_{dl} C$ holds;¹ that is, if there exists \mathcal{I} such that $C^{\mathcal{I}}$ is non-empty and such that $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ holds for each $C_1 \sqsubseteq C_2$ in \mathcal{T} .

2.1 Symbolic Model Checking

Definition 2 (Kripke Structure) *Let V be a set of Boolean variables. A Kripke structure M over V is a four tuple $M = (S, I, R, L)$ where*

1. S is a finite set of states.
2. $I \subseteq S$ is the set of initial states.
3. $R \subseteq S \times S$ is a transition relation that must be total, that is, for every state $s \in S$ there is a state $s' \in S$ such that $R(s, s')$.
4. $L : S \rightarrow 2^V$ is a function that labels each state with the set of variables true in that state.

We view each state s as a truth assignment to the variables V . We view a set of states as a Boolean function over V , characterizing the set. For example, The set of initial states I is considered as a Boolean function over V . Thus, if a state s belongs to I , we write $s \models I$. Similarly, if $v_i \in L(s)$ we write $s \models v_i$, and if $v_i \notin L(s)$ we write $s \models \neg v_i$. We say that $w = s_0, s_1, \dots, s_k$ is a path in M if $\forall i, 0 \leq i < k, (s_i, s_{i+1}) \in R$ and $s_0 \models I$.

In practice, the full Kripke structure of a system is not explicitly given. Rather, a model is given as a set of Boolean variables $V = \{v_1, \dots, v_n\}$, their initial values and their next-state assignments. The definition we give below is an abstraction of the input language of *SMV* [10].

Definition 3 (Model Description) *Let $V = \{v_1, \dots, v_n\}$ be a set of Boolean variables. A tuple $MD = (I_{MD}, [(c_1, c'_1), \dots, (c_n, c'_n)])$ is a Model Description over V where I_{MD}, c_i, c'_i are Boolean expressions over V .*

The semantics of a model description is a Kripke structure $M_{MD} = (S, I_M, R, L)$, where $S = 2^V$, $L(s) = s$, $I_M = \{s \mid s \models I_{MD}\}$, and $R = \{(s, s') : \forall 1 \leq i \leq n, s \models c_i \text{ implies } s' \models \neg v_i \text{ and } s \models c'_i \wedge \neg c_i \text{ implies } s' \models v_i\}$.

Intuitively, a pair $\langle c_i, c'_i \rangle$ defines the next-state assignment of variable v_i in terms of the current values of $\{v_1, \dots, v_n\}$. That is,

$$\text{next}(v_i) = \begin{cases} 0 & \text{if } c_i \\ 1 & \text{if } c'_i \wedge \neg c_i \\ \{0, 1\} & \text{otherwise} \end{cases}$$

where the assignment $\{0, 1\}$ indicates that for every possible next-state value of variables $v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n$ there must exist a next-state with $v_i = 1$, and a next-state with $v_i = 0$.

¹ We write “ \models_{dl} ” to distinguish the use of the double turnstyle symbol by both description logic and model checking communities.

Safety Formulas The formulas we consider are *safety* formulas, given as $AG(b)$ in *CTL* [7], or $G(b)$ in *LTL* [12]. Such formulas state that the Boolean expression b holds on all reachable states of the model under verification. We note that a large and useful subset of *CTL* and *LTL* can be translated into $AG(b)$ type formulas [2].

Bounded Model Checking Given a Kripke structure $M = (S, I, R, L)$, a formula φ , and a bound k , Bounded Model Checking (BMC) tries to refute $M \models \varphi$ by proving the existence of a witness to the negation of φ of length k or less. For $\varphi = AG(b)$, we say that $M^k \not\models \varphi$ if and only if there exists a path $w = s_0, \dots, s_j$ in M such that $j \leq k$ and $s_j \models \neg b$.

2.2 Bounded Model Checking Using Description Logic Reasoning

We briefly describe how bounded model checking can be achieved using description logic reasoning. For a detailed explanation and proof of correctness, refer to [3].

Let $MD = (I, [\langle c_1, c'_1 \rangle, \dots, \langle c_n, c'_n \rangle])$ be a model description for the model $M_{MD} = (S, I, R, L)$, over $V = \{v_1, \dots, v_n\}$. Let k be the bound and let $\varphi = AG(b)$ be a safety formula. We generate a terminology \mathcal{T}_{MD}^k , linear in the size of MD , and a concept C_φ such that $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$ if and only if $M_{MD}^k \not\models \varphi$.

For each variable $v_i \in V$ we introduce one primitive concept V_i , where V_i denotes $v_i = 1$ and $\neg V_i$ denotes $v_i = 0$. We introduce one primitive role R corresponding to the transition relation of the model. We define the concept S_0 to represent I , by replacing each v_i in I with the concept V_i , and the connectives \wedge, \vee, \neg with \sqcap, \sqcup, \neg . The concepts C_i, C'_i correspond to the Boolean conditions c_i, c'_i in the same way. We then introduce three types of concept inclusions:

1. (*inclusions describing the model*) For each pair $\langle c_i, c'_i \rangle$ add the pair of inclusions

$$\begin{aligned} C_i &\sqsubseteq \forall R. \neg V_i \\ (\neg C_i \sqcap C'_i) &\sqsubseteq \forall R. V_i. \end{aligned}$$

2. (*inclusions describing sets of reachable states, of distance i from the initial set*) For a bound k , add k primitive concepts, S_1, \dots, S_k , and for $1 \leq i \leq k$, add the k inclusions

$$S_i \sqsubseteq \exists R. \neg. S_{i-1}.$$

3. (*inclusion to describe the specification*) Let $\varphi = AG(b)$ be the specification to be verified. The Boolean formula b is translated to a concept B in the usual way; in particular, each variable v_i is mapped to the concept V_i , and the Boolean connectives \vee, \wedge, \neg into their corresponding concept constructors \sqcup, \sqcap, \neg .

We define the concept $C_\varphi \equiv \neg B \sqcap (S_0 \sqcup S_1 \sqcup \dots \sqcup S_k)$. If C_φ is consistent with respect to the terminology \mathcal{T}_{MD}^k then $\neg b$ must hold in some state with distance less than k from the initial state. Verification is therefore reduced to the query: $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$.

Theorem 4 (from [3]). $M_{MD}^k \not\models \varphi$ if and only if $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$.

Let $|\mathcal{T}_{MD}^k|$ represent the number of concept inclusions in \mathcal{T}_{MD}^k , and let n be the number of state variables in the model M_{MD} . The following proposition is discussed in [3].

Proposition 5. (size of translation) $|\mathcal{T}_{MD}^k| = 2 \cdot n + k + 1$.

3 On Controlling Propositional vs. Modal Reasoning

The above translation of a BMC problem into a terminology, denoted \mathcal{T}_{MD}^k , uses one set of primitive concepts corresponding to the state variables, and k concepts S_1, \dots, S_k , where S_i represents the set of states of distance i from the initial state. Thus, reaching a state that is i steps from an initial state will require a DL reasoner to build an R -chain consisting of i nodes. On the suspicion that reducing the length of this chain might improve performance, we generalize the above terminological embedding of model descriptions with an ability to supply an additional parameter l so that the resulting terminology, denoted $\mathcal{T}_{MD}^{k/l}$ would entail a reduction of the length of the chain by a factor of l . We present the details of this more elaborate embedding in the remainder of this section. For simplicity, we assume that the original bound k is divisible by l .

The initial set S_0 does not change, and corresponds to I as before. If $k/l > 1$, we introduce a role R . For each variable $v_i \in \{v_1, \dots, v_n\}$, we introduce l primitive concepts V_i^0, \dots, V_i^{l-1} , and for each pair $\langle c_i, c'_i \rangle$, we introduce $2 * l$ concept inclusions of the following form.

$$\begin{aligned} C_i^0 &\sqsubseteq \neg V_i^1 \\ (\neg C_i^0 \sqcap C_i^{0'}) &\sqsubseteq V_i^1 \\ &\vdots \\ C_i^{l-2} &\sqsubseteq \neg V_i^{l-1} \\ (\neg C_i^{l-2} \sqcap C_i^{l-2'}) &\sqsubseteq V_i^{l-1} \\ C_i^{l-1} &\sqsubseteq \forall R. \neg V_i^0 \\ (\neg C_i^{l-1} \sqcap C_i^{l-1'}) &\sqsubseteq \forall R. V_i^0 \end{aligned}$$

If $k/l = 1$ there is no need for the role R , and the last pair of concept inclusions would therefore be omitted. We introduce the concepts S_l, S_{2*l}, \dots, S_k , and the following k/l concept inclusions.

$$\begin{aligned} S_l &\sqsubseteq \exists R^- . S_0 \\ S_{2*l} &\sqsubseteq \exists R^- . S_l \\ &\vdots \\ S_k &\sqsubseteq \exists R^- . S_{k-l} \end{aligned}$$

For a specification $\varphi = AG(b)$, let B^j be the correspondent Boolean expression over the concepts V_i^j for all $0 \leq j < l$. We then define the concept

$$C_\varphi^{0..l-1} \equiv (\neg B^0 \sqcup \dots \sqcup \neg B^{l-1}) \sqcap (S_0 \sqcup S_l \sqcup S_{2*l} \sqcup \dots \sqcup S_k).$$

Proposition 6. (Size of encoding) $|\mathcal{T}_{MD}^{k/l}| = 2 \cdot n \cdot l + k/l + 1$.

The following proposition relates the terminology \mathcal{T}_{MD}^k to the reduced modality terminology $\mathcal{T}_{MD}^{k/l}$.

Proposition 7. $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$ if and only if $\mathcal{T}_{MD}^{k/l} \models_{dl} C_\varphi^{0..l}$.

Proof (sketch). (\implies) Suppose there exists an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for $\mathcal{T}_{MD}^k \models_{dl} C_\varphi$. Since C_φ is not empty in this interpretation, there must exist a j such that $S_j^{\mathcal{I}}$ is not

empty. Let $\sigma_j \in S_j^{\mathcal{I}}$. Since the concept inclusion $S_j \sqsubseteq \exists R^- . S_{j-1}$ holds in the interpretation, and $S_j^{\mathcal{I}}$ is not empty, we deduce that $S_{j-1}^{\mathcal{I}}$ is not empty, and that $\exists \sigma_{j-1} \in S_{j-1}^{\mathcal{I}}$, such that $(\sigma_{j-1}, \sigma_j) \in R^{\mathcal{I}}$. By similar considerations, there must exist a sequence of elements $\sigma_0, \dots, \sigma_j \in \Delta^{\mathcal{I}}$, such that for $0 \leq i < j$, $(\sigma_i, \sigma_{i+1}) \in R^{\mathcal{I}}$, and $\sigma_0 \in S_0^{\mathcal{I}}$. We build an interpretation $\mathcal{I}_l = (\Delta^{\mathcal{I}_l}, \cdot^{\mathcal{I}_l})$ for $\mathcal{T}_{MD}^{k/l}$. $\Delta^{\mathcal{I}_l}$ will consist of $j/l + 1$ elements, $\gamma_0, \dots, \gamma_{j/l}$, where each γ_i corresponds to l consequent elements from $\Delta^{\mathcal{I}}$. The mapping $\cdot^{\mathcal{I}_l}$ will be defined according to $\cdot^{\mathcal{I}}$. Thus, $\forall 1 \leq i \leq n, \forall 0 \leq j < l, \gamma_0 \in V_i^j$ if and only if $\sigma_j \in V_i$. In a similar manner, γ_1 will be mapped according to $\sigma_l, \dots, \sigma_{2 \cdot l - 1}$ and $\gamma_{k/l}$ according to $\sigma_{k-l}, \dots, \sigma_j$. It remains to show that the concept inclusions of $\mathcal{T}_{MD}^{k/l}$ hold under the interpretation \mathcal{I}_l , and that the interpretation of $C_\varphi^{0..l}$ is not empty. These follow easily from the definitions, given that all concept inclusions of \mathcal{T}_{MD}^k hold under \mathcal{I} .

The opposite direction proceeds in a similar way. □

Example Consider the model description

$$\text{Exmp} = (I, [\langle v_1 \wedge v_2, v_3 \rangle, \langle \neg v_2, v_1 \wedge \neg v_1 \rangle, \langle \neg v_1, v_1 \rangle])$$

over $V = \{v_1, v_2, v_3\}$ with $I = \neg v_1 \wedge v_2 \wedge \neg v_3$. Figure 1 draws the states and transitions of the Kripke structure M_{Exmp} described by Exmp , where the label of each state is the value of the vector (v_1, v_2, v_3) . Let the formula to be verified be $\varphi = AG(\neg v_2 \vee$

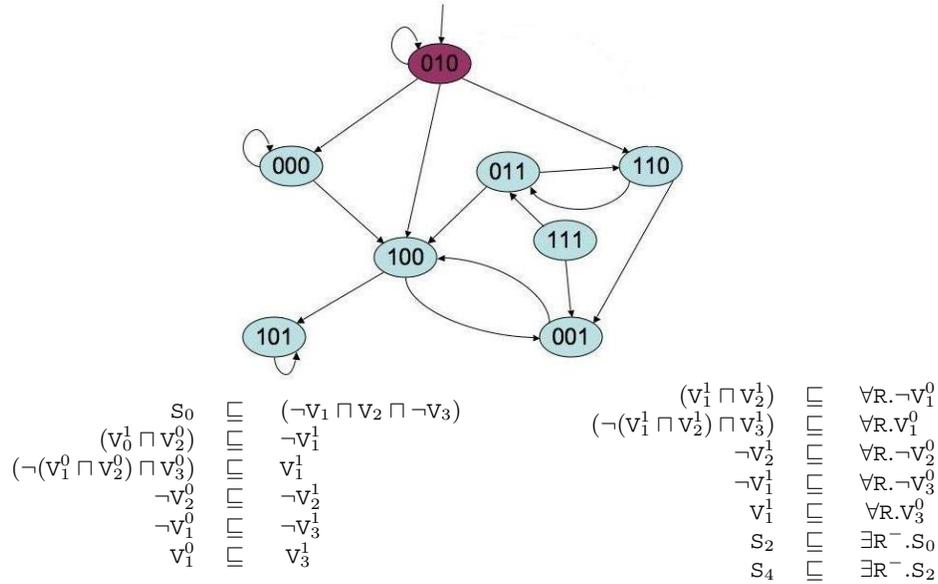


Fig. 1. Kripke Structure and Terminology for “Exmp”

$\neg v_3)$. Note that $M_{\text{Exmp}} \not\models \varphi$, as can be seen in Figure 1, since the state $(0, 1, 1)$, that

contradicts φ , can be reached in two steps from the initial state. We choose the bound to be $k = 4$, and the reduction factor 2.

We build the terminology $\mathcal{T}_{\text{Exmp}}^{4/2}$ for Exmp . We introduce one primitive role R and two sets of primitive concepts: V_1^0, V_2^0, V_3^0 and V_1^1, V_2^1, V_3^1 . The initial state, represented by the concept S_0 , depends only on the set V_i^0 : $S_0 \sqsubseteq (\neg V_1^0 \sqcap V_2^0 \sqcap \neg V_3^0)$. The rest of $\mathcal{T}_{\text{Exmp}}^{4/2}$ is composed of the transition relation of the model, as given in Figure 1. For the specification $\varphi = AG(\neg v_2 \vee \neg v_3)$ we have $B^0 \equiv \neg V_2^0 \sqcup \neg V_3^0$ and $B^1 \equiv \neg V_2^1 \sqcup \neg V_3^1$. The concept $C_\varphi^{0..1}$ is then defined as $C_\varphi^{0..1} \equiv (\neg B^0 \sqcup \neg B^1) \sqcap (S_0 \sqcup S_2 \sqcup S_4)$.

Verification is carried out by asking the query: $\mathcal{T}_{\text{Exmp}}^{4/2} \models_{dl} C_\varphi^{0..1}$.

4 Evaluation

We conducted an experiment on a model derived from the NuSMV example “dme1-16”, taken from [11], to test our hypothesis that reducing the number of nodes created during model building would improve performance when using a DL system for bounded model checking. The original model from [11] was composed of 16 symmetric “cells”, each consisting of 17 propositional variables. We reduced the model to have only 2 cells, in order to get a reasonable run-time. The formula verified expresses a safely condition that is satisfied in the model. We used the DL reasoner *FaCT++* [9], and as expected according to our translation, all runs returned an “unsatisfiable” result. Table 4 reports on the length of time required to determine this for reduction factors of 1 (no reduction), 2 and 4. We believe the times are clear evidence that what really happens is contrary to our hypothesis.

Table 1. Modal vs. Propositional Reasoning

Variables	Bound (k)	Reduction Factor	Time (m)
34	8	1	140
34	8	2	197
34	8	4	686

4.1 Discussion

Highly successful SAT solvers such as Minisat [8] use model building algorithms that operate by progressively refining an understanding of a “possible world”. DL systems such as *FaCT++* also use model building algorithms, but, to relate their behavior to typical SAT solvers, must deal with the added complication of modal reasoning in which a potentially large number of possible worlds are involved. For applications such as fixed depth model checking, DL systems therefore enable a tradeoff between the complexity of particular worlds and the number of worlds.

We were quite surprised that moving towards fewer but more complicated worlds would have the negative impact on performance that it did in our experiment, which prompted a lot of reflection on why this happens. We conclude with some suggestions on directions of future research on how DL technology might be adapted in order to improve its performance on applications like model checking.

Part of this reflection was to conduct a small literature survey on how modern SAT solvers and DL reasoners are implemented [1, 8]. It became quickly apparent that, e.g.,

Minisat relies heavily on using arrays to encode knowledge about a particular world, and that DL technology is more likely to encode similar knowledge in separate records, and to navigate among the records via pointers. It is common folklore that, when feasible, replacing pointer navigation with array indexing will improve the performance of algorithms, which suggests one possible avenue for improving the performance of propositional reasoning in DL systems.

A non-trivial problem for DL systems relates to *blocking*. In particular, such systems must frequently compare different possible worlds to ensure that model building will terminate. This prompted a more careful consideration of the structure of the terminology encoding a model description. We noticed that it might be straightforward to recognize that the “schema” underlying all occurrences of the “ $\exists R^- . C$ ” concept constructor was acyclic, which suggests a possible extension to, e.g., preprocessing in *FaCT++* in which such acyclic (sub)schema are recognized, and consequently that blocking activity during model building is disabled for possible worlds “within” acyclic schema. Indeed, any terminology generated by our reductions will always satisfy a global acyclicity property that would allow disabling any processing relating to blocking.

Acknowledgements

We thank Dmitry Tsarkov for his support in the installation of *FaCT++*, and Peter Tarle and Nortel for many valuable discussions on this work and for financial support. The authors are also supported in part by grants from NSERC of Canada.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. I. Beer, S. Ben-David, and A. Landver. On-the-fly model checking of RCTL formulas. In *Proc. 10th International Conference on Computer Aided Verification (CAV'98)*, LNCS 1427, pages 184–194. Springer-Verlag, 1998.
3. S. Ben-David, R. Trefler, and G. Weddell. Bounded model checking with description logic reasoning. In *Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, pages 60–72, July 2007.
4. A. Biere, A. Cimatti, E. Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *TACAS'99*, 1999.
5. Randy Bryant. Graph-based algorithms for boolean function manipulation. In *IEEE Transactions on Computers*, volume c-35 no. 8.
6. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 2000.
7. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logics of Programs*, LNCS 131, pages 52–71. Springer-Verlag, 1981.
8. Niklas Een and Niklas Sorensson. An Extensible SAT-solver. In *Theory and Applications of Satisfiability Testing*, pages 502–518. Springer Berlin/Heidelberg, LNCS 2919, 2004.
9. I. Horrocks. The FaCT system. pages 307–312, 1998.
10. K. McMillan. *Symbolic model checking*, 1993.
11. NuSMV examples collection. <http://nusmv.irst.itc.it/examples/examples.html>.
12. Amir Pnueli. The temporal logic of programs. In *18th IEEE Symposium on Foundation of Computer Science*.
13. J. Quielle and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming*, 1982.