Bounded Model Checking with Description Logic Reasoning

Shoham Ben-David, Richard Trefler, Grant Weddell

David R. Cheriton School of Computer Science University of Waterloo
Technical Report CS-2007-07

March 26, 2007

Abstract. Model checking is a technique for verifying that a finite-state concurrent system is correct with respect to its specification. In *bounded* model checking (BMC), the system is unfolded until a given depth, and translated into a CNF formula. A SAT solver is then applied to the CNF formula, to find a satisfying assignment. Such a satisfying assignment, if found, demonstrates an error in the model of the concurrent system.
Description Logic (DL) is a family of knowledge representation formalisms, for which reasoning is based on tableaux techniques. We show how Description Logic can serve as a natural setting for representing and solving a BMC problem. We formulate a bounded model checking problem as a consistency problem in the DL dialect $\mathcal{ALCI}$. Our formulation results in a compact representation of the model, one that is linear in the size of the model description, and does not involve any unfolding of the model. Experimental results, using the DL reasoner FaCT++, significantly improve on a previous approach that used DL reasoning for model checking.

1 Introduction

Model checking ([8, 20], c.f.[9]) is a technique for verifying finite-state concurrent systems, that has been proven to be very effective in the verification of 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. Essentially, verification of the formula φ on a model M, checks that the tree of all computations of M satisfies φ.

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 variables describing it. 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, known as *SMV* [17], is based on Binary Decision Diagrams (BDDs) [6] for representing the state space as well as for performing the model checking

procedure. The second is known as Bounded Model Checking (*BMC*) [5]. Using this method, the model under verification is unfolded k times (for a given bound k), and translated into a propositional CNF formula. 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.

Description Logic (DL) ([2]) is a family of knowledge representation formalisms mainly used for specifying ontologies for information systems. An ontology $\mathcal{T}$ is called a *terminology* or more simply a *Tbox*, and corresponds to a set of concept inclusion dependencies. Each inclusion dependency has the form $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$, and asserts containment properties of relevant concepts in an underlying domain, e.g., that *managers* are included in *employees*

$$\text{MANAGER} \sqsubseteq \text{EMPOYEE},$$

and also in *those things that hire only employees*

$$\text{MANAGER} \sqsubseteq \forall \textit{hires}.\text{EMPLOYEE}.$$

In this latter case, *hires* is an example of a *role*. In DLs, a role is always a binary relation over the underlying domain.

The main reasoning service provided by a DL system is *concept consistency*; that is, for a given terminology $\mathcal{T}$ and concept $\mathtt{C}$, to determine if there is a non-empty interpretation of the concept that also satisfies each inclusion dependency in $\mathcal{T}$, written $\mathcal{T} \models_{dl} \mathtt{C}$. Most DL systems implement this service by employing some form of tableaux or model building techniques. The examples illustrate that these techniques manifest both propositional and modal reasoning (where *hires* is viewed as an event), which makes using a DL system an attractive possibility for model checking.

To explore this, we consider a goal directed embedding of BMC problems as concept consistency problems in the DL dialect $\mathcal{ALCI}$. Our encoding of a model description as a terminology in $\mathcal{ALCI}$ results in a natural *symbolic* representation of the sets of states and state transitions. Specifically, given a model description MD and a bound k, we formulate a *BMC* problem as a terminology $\mathcal{T}^k_{MD}$ over $\mathcal{ALCI}$. Our formulation is compact, and does not involve *unfolding* of the model. Rather, the size of the terminology is the same as the size of the description of the model plus a set of k concept inclusions that are needed for the bounded verification. In contrast, the known *BMC* method that uses a SAT solver for this task needs k copies of the model description. This produces a representation that is k times larger than ours. For simplicity, we assume the formula to be verified expresses a *safety* property (an $AG(b)$ type formula), although more complex formulas can also be supported.

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

$$\mathsf{C}_1 \sqsubseteq \forall \mathsf{R}.\mathsf{C}_2$$

stating that if the current state satisfies the condition represented by C_1, then all the next-states that can be reached in one step by R, must satisfy the condition C_2. Note that interpretations for this set of concept inclusions correspond to sub-models of the given model M.

Let the concept S_0 represent the set of initial states of M. If S_1 represents states that can be reached in one step from S_0, then the concept inclusion $\mathsf{S}_1 \sqsubseteq \exists \mathsf{R}^-.\mathsf{S}_0$ must hold (that is, the set S_1 is a subset of all the states that can reach S_0 by going one step backwards using the relation R). Similarly, we denote by S_i subsets of the states reachable in k steps from the set of initial states, and introduce the inclusions

$$\mathsf{S}_i \sqsubseteq \exists \mathsf{R}^-.\mathsf{S}_{i-1}$$

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 representing the state variables). Model checking is now carried out by asking the query: "does there exist an interpretation for the above set of concept inclusions, such that $\mathsf{C}_\varphi (= \neg \mathsf{B} \sqcap \mathsf{S}_i)$ is not empty for some S_i?". A positive answer from the DL reasoner indicates an error in M.

We relate the consistency of the concept C_φ with respect to the terminology $\mathcal{T}_M^k$ to the satisfaction of φ in the model M, by proving that $M_M^k \not\models \varphi$ if and only if $\mathcal{T}_M^k \models_{dl} \mathsf{C}_\varphi$ is consistent.

Note that this formulation of a model checking problem is *goal directed*. That is, the DL reasoner begins from a description of buggy states ($\neg \mathsf{B} \sqcap \mathsf{S}_i$), and proceeds from there to find a legal backward path to a description of initial states. In earlier preliminary work using a DL for model checking [4], we explored a synchronous forward reasoning approach. In comparison to this earlier approach, our experimental results confirm that goal directed encodings perform far better, indeed outperforming a BDD-based technology for the sample safety property considered. However, the combination of our current encoding and current DL reasoning technology [14] is still not competitive with SAT-based approaches. We give some suggestions for future work to address this in our concluding remarks.

The rest of the paper is organized as follows. The next section provides the necessary background definitions. Our main contributions then follow in Section 3 in which we formally define our translation and prove its correctness,

and in which we report on some preliminary experimental results. In Section 4 we discuss related work. Summary comments and conclusions then follow in Section 5.

2 Background and Definitions

2.1 Description Logic

Description Logics [2] come in different dialects. The basic DL dialect is called *Attributive Language* with *Complements*, or $\mathcal{ALC}$. For our purposes we need the more expressive dialect $\mathcal{ALCI}$, allowing the use of role inverse. Its definition is given below.

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

- *If* $R_1 \in$ R *then so is* R_1^-.

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* $\exists R.C \in$ C.

Additional concepts are defined as syntactic sugaring of those above:

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

An *inclusion dependency* is an expression of the form $C_1 \sqsubseteq C_2$. A *terminology* $\mathcal{T}$ 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.

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

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* $\mathcal{ALCI}$ 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 $\mathcal{T}$.

2.2 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. *$L : S \to 2^V$ 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 $w = s_0, s_1, ..., s_k$ 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 $V = \{v_1, ..., v_n\}$, their initial values and their next-state assignments. The definition we give below is an abstraction of the input language of *SMV* [17].

Definition 3 (Model Description) *Let $V = \{v_1, ..., v_n\}$ be a set of Boolean variables. A tuple $MD = (I_{MD}, [\langle c_1, c_1' \rangle, ..., \langle c_n, c_n' \rangle])$ 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 $S = 2^V$, $L(s) = s$, $I_M = \{s | s \models I_{MD}\}$, and $R = \{(s, s') : \forall 1 \leq i \leq n,\ s \models c_i$ implies $s' \models \neg v_i$ and $s \models c_i' \wedge \neg c_i$ 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 $\{v_1, ..., v_n\}$. That is,

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

[1] In the DL worlds, the sign $\models$ is used to indicate consistency. However, this same sign is used also in model checking to indicate a formula is satisfied in a model. We therefore use $\models_{dl}$ to indicate consistency in DL.

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

Safety Formulas The formulas we consider are *safety* formulas, given as $AG(b)$ in CTL [8], or $G(b)$ in LTL [19]. 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 [3].

Bounded Model Checking Given a Kripke structure M, 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 $w = s_0, ..., s_j$, such that $j \leq k$ and $s_j \models \neg b$.

The original BMC method [5] generates a propositional formula that is satisfiable if and only if $M^k \not\models \varphi$. We show how to achieve this using Description Logic.

3 Bounded Model Checking using Description Logic

We give a linear reduction of a bounded model checking problem into a consistency check over $\mathcal{ALCI}$. Our method performs bounded reachability on the given model, and thus resembles the BMC [5] method. However, classical BMC methods unfold the model k times (for a bound k), introducing k copies of the state variables, as well as the transition relation. Our method in contrast, uses only *one* copy of each state variable, and defines reachability of bound k as a set of k concept inclusions. Thus our method resembles the reachability algorithm performed in BDD-based symbolic model checking [17]. Our method can therefore be seen as a combination of the two major approaches currently existing for symbolic model checking.

In the next section we present the translation into a DL terminology. We demonstrate the translation using an example in section 3.2, and then prove the correctness of the translation in section 3.3. In section 3.4 we discuss implementation and experimental results.

3.1 Constructing a Terminology over $\mathcal{ALCI}$

Let $MD = (I, [\langle c_1, c'_1\rangle, ..., \langle c_n, c'_n\rangle])$ be a model description for the model $M_{MD} = (S, I, R, L)$, over $V = \{v_1, ..., v_n\}$. Let k be the bound and let φ

be a safety formula. We generate a terminology $\mathcal{T}^k_{MD}$, linear in the size of MD, and a concept C_φ, such that $\mathcal{T}^k_{MD} \models C_\varphi$ is consistent if and only if $M^k_{MD} \not\models \varphi$.

For each variable $v_i \in V$ we introduce one primitive concept V_i, where V_i denotes $v_i = 1$ and $\neg\mathsf{V}_i$ denotes $v_i = 0$. We introduce one primitive role R corresponding to the transition relation of the model.

We construct the terminology $\mathcal{T}^k_{MD}$ as the union of two terminologies: $\mathcal{T}^k_{MD} = \mathcal{T}_{MD} \cup \mathcal{T}_k$, where the terminology $\mathcal{T}_{MD}$ depends on the model description, and $\mathcal{T}_k$ depends only on the bound k of the number of cycles searched. The construction of $\mathcal{T}_{MD}$ and $\mathcal{T}_k$ are given below.

Constructing $\mathcal{T}_{MD}$ Let $MD = (I, [\langle c_1, c'_1\rangle, ..., \langle c_n, c'_n\rangle])$ be a model description, where I, c_i, c'_i are Boolean expressions over the variables $V = \{v_1, ..., v_n\}$. 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 $\sqcap, \sqcup, \neg$.

Let the pair $\langle c_i, c'_i\rangle$ describe the next state behavior of the variable v_i. 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 $\{0, 1\}$ is a non-deterministic assignment, allowing v_i to assume both 0 and 1 in the next state. Let C_i be the concept generated by replacing every v_i in c_i with the concept V_i, and $\wedge$ with $\sqcap$. Let C'_i be the concept corresponding to c'_i in the same way. We introduce the following concept inclusions.

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

In total, two concept inclusions are introduced for each variable v_i in MD (corresponding to the pair $\langle c_i, c'_i\rangle$).

Constructing $\mathcal{T}_k$. For a bound k, we introduce k primitive concepts, $\mathsf{S}_1, ..., \mathsf{S}_k$. For $1 \leq i \leq k$, we introduce k inclusions:

$$\mathsf{S}_i \sqsubseteq \exists\mathsf{R}^-.\mathsf{S}_{i-1},$$

Note that the concept inclusions in $\mathcal{T}_k$ are purely syntactic and do not depend on the model description under verification MD. In fact, the same set of inclusions shall appear in the verification (of bound k) of any model.

Constructing C_φ. Let φ be the specification to be verified. As mentioned before, we are concerned with safety formulas, asserting "$AG(b)$", with b being a Boolean formula over the variables $v_1, ..., v_n$. To show that such a formula does not hold, it is enough to find one state s of the Kripke structure, reachable from the initial state, such that $s \models \neg b$. We translate the Boolean formula b into a concept B in the usual way, where each variable v_i is translated to the concept V_i, and the Boolean connectives $\vee, \wedge$ into their correspondents $\sqcup, \sqcap$.

We define the concept $\mathsf{C}_\varphi \equiv \neg\mathsf{B} \sqcap (\mathsf{S}_0 \sqcup \mathsf{S}_1 \sqcup ... \sqcup \mathsf{S}_k)$. If C_φ is consistent with respect to the terminology $\mathcal{T}^k_{MD} = \mathcal{T}_k \cup \mathcal{T}_{MD}$ it means that $\neg b$ holds in some state, with distance less than k from the initial state. Verification is therefore reduced to the query: $\mathcal{T}^k_{MD} \models_{dl} \mathsf{C}_\varphi$.

3.2 Example

Consider the model description

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

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 $M_{\mathtt{Exmp}}$ 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

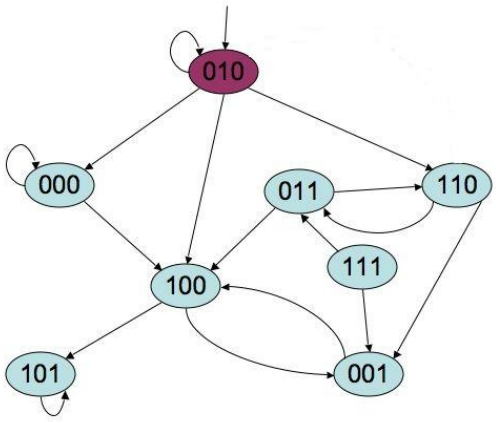

Fig. 1. A Kripke structure for Exmp

be $\varphi = AG(\neg v_2 \vee \neg v_3)$. Note that $M_{\mathtt{Exmp}} \not\models \varphi$, 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 $k = 4$.

In order to build a terminology for Exmp we introduce one primitive role R and three primitive concepts $\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3$. We first build the terminology $\mathcal{T}_{\mathsf{Exmp}}$. For the initial state, represented by the concept S_0, we introduce the following concept inclusion:

$$\mathsf{S}_0 \sqsubseteq (\neg\mathsf{V}_1 \sqcap \mathsf{V}_2 \sqcap \neg\mathsf{V}_3)$$

The rest of $\mathcal{T}_{\mathsf{Exmp}}$ is composed of the transition relation of the model, as given below.

$$\begin{aligned}
(\mathsf{V}_1 \sqcap \mathsf{V}_2) &\sqsubseteq \forall\mathsf{R}.\neg\mathsf{V}_1 \\
(\neg(\mathsf{V}_1 \sqcap \mathsf{V}_2) \sqcap \mathsf{V}_3) &\sqsubseteq \forall\mathsf{R}.\mathsf{V}_1 \\
\neg\mathsf{V}_2 &\sqsubseteq \forall\mathsf{R}.\neg\mathsf{V}_2 \\
\neg\mathsf{V}_1 &\sqsubseteq \forall\mathsf{R}.\neg\mathsf{V}_3 \\
\mathsf{V}_1 &\sqsubseteq \forall\mathsf{R}.\mathsf{V}_3
\end{aligned}$$

Note that for simplicity, we omitted the inclusion $(\neg\neg\mathsf{V}_2 \sqcap \mathsf{V}_1 \sqcap \neg\mathsf{V}_1) \sqsubseteq \forall\mathsf{R}.\mathsf{V}_2$ (corresponding to $\neg\mathsf{C}_i \sqcap \mathsf{C}'_i \sqsubseteq \forall\mathsf{R}.\mathsf{V}_i$ for $i = 2$), since the prefix $\neg\neg\mathsf{V}_2 \sqcap \mathsf{V}_1 \sqcap \neg\mathsf{V}_1$ is actually equivalent to $\bot$. Similarly, the concept $\neg\neg\mathsf{V}_1 \sqcap \mathsf{V}_1$ (corresponding to $\neg\mathsf{C}_3 \sqcap \mathsf{C}'_3$) was replaced by the equivalent V_1.

In order to "unfold" the model four times (for the chosen bound $k = 4$), we introduce the primitive concepts $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3, \mathsf{S}_4$, and the concept inclusions:

$$\begin{aligned}
\mathsf{S}_1 &\sqsubseteq \exists\mathsf{R}^-.\mathsf{S}_0 \\
\mathsf{S}_2 &\sqsubseteq \exists\mathsf{R}^-.\mathsf{S}_1 \\
\mathsf{S}_3 &\sqsubseteq \exists\mathsf{R}^-.\mathsf{S}_2 \\
\mathsf{S}_4 &\sqsubseteq \exists\mathsf{R}^-.\mathsf{S}_3
\end{aligned}$$

For the specification $\varphi = AG(\neg v_2 \vee \neg v_3)$ we get $\mathsf{B} \equiv \neg\mathsf{V}_2 \sqcup \neg\mathsf{V}_3$, and $\mathsf{C}_\varphi \equiv \neg\mathsf{B} \sqcap (\mathsf{S}_0 \sqcup \mathsf{S}_1 \sqcup \mathsf{S}_2 \sqcup \mathsf{S}_3 \sqcup \mathsf{S}_4)$. We can now present the full terminology $\mathcal{T}^k_{\mathsf{Exmp}}$, as shown in Figure 2 below. Verification is then carried out by asking the query: Is the concept C_φ consistent with respect to $\mathcal{T}^4_{\mathsf{Exmp}}$?
In the next section we prove the correction of our translation.

3.3 Correctness

We relate the consistency of the concept C_φ with respect to $\mathcal{T}^k_{MD}$ to the satisfaction of φ in the model M_{MD}. Let $MD = (I, [\langle c_1, c'_1\rangle, ..., \langle c_n, c'_n\rangle])$ denote a model description for a model $M_{MD} = (S, I_M, R, L)$, and let $\varphi = AG(b)$ be a safety formula. Let $\mathcal{T}^k_{MD}$ be the terminology built for MD, as defined in section 3.1, and let C_φ be the concept representing φ.

Theorem 4. *$M^k_{MD} \not\models \varphi$ if and only if $\mathcal{T}^k_{MD} \models_{dl} \mathsf{C}_\varphi$ is consistent.*

S_0	$\sqsubseteq$	$(\neg\mathsf{V}_1 \sqcap \mathsf{V}_2 \sqcap \neg\mathsf{V}_3)$
$(\mathsf{V}_1 \sqcap \mathsf{V}_2)$	$\sqsubseteq$	$\forall\mathsf{R}.\neg\mathsf{V}_1$
$(\neg(\mathsf{V}_1 \sqcap \mathsf{V}_2) \sqcap \mathsf{V}_3)$	$\sqsubseteq$	$\forall\mathsf{R}.\mathsf{V}_1$
$\neg\mathsf{V}_2$	$\sqsubseteq$	$\forall\mathsf{R}.\neg\mathsf{V}_2$
$\neg\mathsf{V}_1$	$\sqsubseteq$	$\forall\mathsf{R}.\neg\mathsf{V}_3$
V_1	$\sqsubseteq$	$\forall\mathsf{R}.\mathsf{V}_3$
S_1	$\sqsubseteq$	$\exists\mathsf{R}^-.\mathsf{S}_0$
S_2	$\sqsubseteq$	$\exists\mathsf{R}^-.\mathsf{S}_1$
S_3	$\sqsubseteq$	$\exists\mathsf{R}^-.\mathsf{S}_2$
S_4	$\sqsubseteq$	$\exists\mathsf{R}^-.\mathsf{S}_3$

Fig. 2. The terminology $\mathcal{T}^4_{Exmp}$ over $\mathcal{ALCI}$

For the proof of the theorem, we need the following definition and lemma. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for $\mathcal{T}^k_{MD}$. We define a function from the elements of $\mathcal{I}$ to states in S in the following way.

Definition 5 *F from $\mathcal{I}$ to S is a function such that $F(\sigma) = s$ if $\forall 1 \leq i \leq n$, $\sigma \in \mathsf{V}_i$ if and only if $s \models v_i$.*

Note that the function F is well defined, since a state s is determined by the value of the variables $v_1, ..., v_n$.

Lemma 6. *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for $\mathcal{T}^k_{MD}$. Let c be a Boolean expression over $v_1, ..., v_n$, and C its corresponding concept derived by replacing each variable v_i by the concept V_i, and the Boolean connectives $\vee, \wedge, \neg$ by $\sqcup, \sqcap, \neg$. Let $\sigma \in \Delta^{\mathcal{I}}$ be an element in the interpretation $\mathcal{I}$, and let $s = F(\sigma)$. Then $\sigma \in C^{\mathcal{I}}$ if and only if $s \models c$.*

Proof. By induction on the structure of the Boolean expression c. □

Proof. (of Theorem 4)

($\Longrightarrow$) Assume that $M^k_{MD} \not\models \varphi$. Then there exists a path in M^k_{MD}, $w = s_0, ..., s_j$, where $j \leq k$, such that $s_0 \models I$, $\forall 0 < l \leq j, (s_{l-1}, s_l) \in R$, and $s_j \models \neg b$. We build a finite interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ based on w. The set $\Delta^{\mathcal{I}}$ includes $j + 1$ elements $\sigma_0, ..., \sigma_j$. Each of the primitive concepts V_i is interpreted as a set $\mathsf{V}_i^{\mathcal{I}}$, such that $\forall 0 \leq l \leq j$, $\sigma_l \in \mathsf{V}_i^{\mathcal{I}}$ if and only if $s_l \models v_i$. Note that for this interpretation, $F(\sigma_l) = s_l$.

We interpret each primitive concept S_l as $\{\sigma_l\}$ for $0 \leq l \leq j$. The primitive concepts $\mathsf{S}_{j+1}, ..., \mathsf{S}_k$ are interpreted as $\emptyset$. The interpretation $\mathsf{R}^{\mathcal{I}}$ of the role R is a set of pairs (σ_l, σ_{l+1}), $0 \leq l < j$. It remains to show that all concept inclusions of $\mathcal{T}^k_{MD}$ hold under this interpretation, and that $\mathsf{C}^{\mathcal{I}}_{\varphi}$, the interpretation of the concept C_{φ} is not empty.

– Inclusions from $\mathcal{T}_k$: For $l > j$, $\mathsf{S}_l^{\mathcal{I}} = \emptyset$, and are thus included in any other set. In order for $\mathsf{S}_l \sqsubseteq \exists \mathsf{R}^-.\mathsf{S}_{l-1}$ to hold, for $l \leq j$, we need to show that $\mathsf{S}_l^{\mathcal{I}} \subseteq \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} \text{ s.t. } (y,x) \in \mathsf{R}^{\mathcal{I}} \wedge y \in \mathsf{S}_{l-1}^{\mathcal{I}}\}$. Indeed, $\mathsf{S}_l^{\mathcal{I}} = \{\sigma_l\}, \mathsf{S}_{l-1}^{\mathcal{I}} = \{\sigma_{l-1}\}, (\sigma_{l-1}, \sigma_l) \in \mathsf{R}^{\mathcal{I}}$, and (σ_{l-1}, σ_l) is the only pair $(x,y) \in \mathsf{R}^{\mathcal{I}}$ such that $x \in \mathsf{S}_{l-1}^{\mathcal{I}}$. Thus the inclusion holds.
– Inclusions from $\mathcal{T}_{MD}$: We need to show that inclusions of the type $\mathsf{C}_i \sqsubseteq \forall \mathsf{R}.\neg \mathsf{V}_i$ and $\neg \mathsf{C}_i \sqcap \mathsf{C}_i' \sqsubseteq \forall \mathsf{R}.\mathsf{V}_i$, for $1 \leq i \leq n$, hold under the interpretation $\mathcal{I}$. We know that $\forall 0 < l \leq j, (s_{l-1}, s_l) \in R$. According to the definition of model description, it means that $\forall 0 < i \leq n$, $s_{l-1} \models c_i$ implies $s_l \models \neg v_i$ and $s_{l-1} \models \neg c_i \wedge c_i'$ implies $s_l \models v_i$. By lemma 6 we get that $\sigma_{l-1} \in \mathsf{C}_i^{\mathcal{I}}$ implies $\sigma_l \notin \mathsf{V}^{\mathcal{I}}$ and $\sigma_{l-1} \in (\Delta^{\mathcal{I}} \setminus \mathsf{C}_i^{\mathcal{I}}) \cap \mathsf{C}_i'^{\mathcal{I}}$ implies $\sigma_l \in \mathsf{V}^{\mathcal{I}}$. Since no pairs other than (σ_{l-1}, σ_l) belong to $\mathsf{R}^{\mathcal{I}}$ in the interpretation $\mathcal{I}$, the inclusions hold.
– $\mathsf{C}_\varphi^{\mathcal{I}}$ is not empty: We shall show that $\sigma_j \in \mathsf{C}_\varphi^{\mathcal{I}}$. Recall that

$$\mathsf{C}_\varphi \equiv \neg \mathsf{B} \sqcap (\mathsf{S}_0 \sqcup \mathsf{S}_1 \sqcup ... \sqcup \mathsf{S}_k)$$

and therefore under the interpretation $\mathcal{I}$,

$$\mathsf{C}_\varphi^{\mathcal{I}} = (\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}}) \cap (\mathsf{S}_0^{\mathcal{I}} \cup \mathsf{S}_1^{\mathcal{I}} \cup ... \cup \mathsf{S}_k^{\mathcal{I}})$$

Since $\mathsf{S}_j = \{\sigma_j\}$, we get that $\sigma_j \in (\mathsf{S}_0^{\mathcal{I}} \cup \mathsf{S}_1^{\mathcal{I}} \cup ... \cup \mathsf{S}_k^{\mathcal{I}})$. Since $s_j \models \neg b$, we get by Lemma 6 that $\sigma_j \notin \mathsf{B}^{\mathcal{I}}$. Thus $\sigma_j \in \Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}}$, and therefore $\sigma_j \in \mathsf{C}_\varphi^{\mathcal{I}}$.

($\Longleftarrow$) Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation showing that $\mathcal{T}_{MD}^k \models_{dl} \mathsf{C}_\varphi$ is consistent. We have to show that $M_{MD}^k \not\models \varphi$. Since $\mathsf{C}_\varphi^{\mathcal{I}} = (\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}}) \cap (\mathsf{S}_0^{\mathcal{I}} \cup \mathsf{S}_1^{\mathcal{I}} \cup ... \cup \mathsf{S}_k^{\mathcal{I}})$ is not empty in $\mathcal{I}$, it must be the case that for some j, $0 \leq j \leq k$, $(\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}}) \cap \mathsf{S}_j^{\mathcal{I}}$ is not empty. Let σ_j be an element in $(\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}}) \cap \mathsf{S}_j^{\mathcal{I}}$. Then $\sigma_j \in (\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}})$ and also $\sigma_j \in \mathsf{S}_j^{\mathcal{I}}$.

Since $\mathcal{T}_{MD}^k$ includes the concept inclusion $\mathsf{S}_j \sqsubseteq \exists \mathsf{R}^-.\mathsf{S}_{j-1}$, and $\mathsf{S}_j^{\mathcal{I}}$ is not empty, we deduce that $\mathsf{S}_{j-1}^{\mathcal{I}}$ is not empty, and that $\exists \sigma_{j-1} \in \mathsf{S}_{j-1}^{\mathcal{I}}$, such that $(\sigma_{j-1}, \sigma_j) \in \mathsf{R}^{\mathcal{I}}$. By similar considerations, there must exist a sequence of elements $\sigma_0, ..., \sigma_j \in \Delta^{\mathcal{I}}$, such that for $0 \leq l < j$, $(\sigma_l, \sigma_{l+1}) \in \mathsf{R}^{\mathcal{I}}$, and $\sigma_0 \in \mathsf{S}_0^{\mathcal{I}}$. We define a sequence of states $s_0, ..., s_j$ from M_{MD} according to the function F from Definition 5: $F(\sigma_l) = s_l$.

We need to prove that for $0 \leq l < j$, $(s_l, s_{l+1}) \in R$, $s_0 \models I$, $s_j \models \neg b$. These follow easily from Lemma 6 as shown below.

– $s_0 \models I$. Recall that the concept S_0 corresponds to the condition I of the model M_{MD}, which is a Boolean combination of the variables v_i. Thus since $\sigma_0 \in \mathsf{S}_0^{\mathcal{I}}$ we get by Lemma 6 that $s_0 \models I$.

- $s_j \models \neg b$. As shown above, $\sigma_j \in (\Delta^{\mathcal{I}} \setminus \mathsf{B}^{\mathcal{I}})$ and therefore $\sigma_j \notin \mathsf{B}^{\mathcal{I}}$. By Lemma 6, $s_0 \not\models b$, that is $s_j \models \neg b$.
- $(s_l, s_{l+1}) \in R$. Since all concept inclusions of $\mathcal{T}^k_{MD}$ hold under the interpretation $\mathcal{I}$, we know that $\forall 1 \leq i \leq n$,
  $\mathsf{C}_i^{\mathcal{I}} \subseteq \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, (x,y) \in \mathsf{R}^{\mathcal{I}} \rightarrow y \in \Delta^{\mathcal{I}} \setminus \mathsf{V}_i\}$ and also
  $(\Delta^{\mathcal{I}} \setminus \mathsf{C}_i^{\mathcal{I}}) \cup \mathsf{C}_i'^{\mathcal{I}} \subseteq \{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}, (x,y) \in \mathsf{R}^{\mathcal{I}} \rightarrow y \in \mathsf{V}_i\}$.
  Thus if $\sigma_l \in \mathsf{C}_i^{\mathcal{I}}$ it must be the case that $\sigma_{l+1} \notin \mathsf{V}_i^{\mathcal{I}}$, and similarly if $\sigma_l \in (\neg\mathsf{C}_i^{\mathcal{I}} \cap \mathsf{C}_i'^{\mathcal{I}})$ it must be the case that $\sigma_{l+1} \in \mathsf{V}_i^{\mathcal{I}}$. Because of the correspondence between σ_l and s_l, we have that $s_l \models c_i$ implies $s_{l+1} \models \neg v_i$ and $s_l \models c'_i \wedge \neg c_i$ implies $s_{l+1} \models v_i$. Thus by definition, $(s_l, s_{l+1}) \in R$.

This concludes the proof. □

3.4 Experiments

We implemented our method and experimented with it using the Description Logic reasoner *FACT*$^{++}$ [14]. We present two sets of results. In Table 1 we compare our method with the one reported in [4], where a different encoding of model checking in DL is described. The method in [4] applies a *forward*

Table 1. Forward vs. backward model checking using DL

	Forward Search			Backward Search		
Model Size	concepts	inclusions	time	concepts	inclusions	time
10	32	104	0.07	22	25	0
20	62	204	> 1200	32	40	0.01
50	152	514	> 1200	62	110	0.02

search, as opposed to the backward search proposed in this paper. We compare the two methods on a very simple model, parameterized so we can run it with increasing numbers of state variables. For the backward method we chose the bound to be 20. Table 1 shows the number of concepts and concept inclusions needed to describe the model for each method, and the time in seconds it takes to execute. We set a timeout of 1200 seconds. The results demonstrate that the backward search, described in this paper, significantly outperforms the forward search approach.

In Table 2 we present results comparing our method (backward search) to two symbolic model checking tools: *NuSMV* [7] as a BDD-based model checker, and *zChaff* [18] as a SAT solver for bounded model checking. The model we use for comparison is derived from the NuSMV example "dme1-16", taken from [1] and parameterized to have different numbers of cells. The formula verified is a safety one, that holds in the model. We did not attempt to

Table 2. DL Model Checking Vs. SMV and SAT

Model Size	Bound	DL-Backward	NuSMV	zChaff
85	5	4	> 1200	1
85	10	> 1200	> 1200	0.9
272	5	202	> 1200	1.2
272	10	> 1200	> 1200	3.9
425	5	335	> 1200	2.4
425	10	> 1200	> 1200	6.7

optimize the run of any tool (many options are available), but rather, ran them in their default mode. Although our method performs better than NuSMV on the given examples, it still falls far behind the performance of zChaff. In particular we note that the DL method seems to be very sensitive to the bound k on the depth of the search.

4 Related Work

A connection between knowledge-base reasoning and model checking has been explored before. Gottlob et al in [12, 13] analyzed the expressive power of *Datalog* statements, and compared them to known temporal logics. Sahasrabudhe in [21] has performed model checking of telephony feature interactions using SQL, and compared the results with model checking of the same system using the model checker SMV [17]. Both these works however, used an explicit representation of the model, as opposed to the symbolic representation that we propose. This difference is crucial, since in many cases the Kripke structure for the model is too big to be built, and symbolic methods must be used. Dovier and Quintarelli in [11] were interested in the opposite direction: they translated a knowledge-base into a Kripke structure, and a query into a temporal logic formula. They then used a model checker to make inferences about the knowledge-base.

In a previous paper [4] we gave a first formulation of a model checking problem as a terminology in Description logic. The formulation there has some advantages over the current: it performs unbounded model checking rather than bounded model checking that we propose here; it supports safety as well as liveness formulas, and it uses the simpler dialect $\mathcal{ALC}$, rather then $\mathcal{ALCI}$ that we use here. However, the terminology built for a given model description three times as many concepts and five times as many concept inclusions. In addition, the reasoning involved synchronizing the progress of the different state variables. Thus the performance of that method, as shown in section 3.4, was much worse than the one presented in this paper.

Finally, a compact representation of a BMC problem, with size similar to the one described in this paper, is also achieved when presenting the model description as a Quantified Boolean Formula (QBF). Recent activity in this area [10, 16] suggest though, that this too does not perform as well as SAT solvers.

5 Conclusions and Future Work

We have shown how Description Logic can serve as a natural setting for representing a BMC problem, avoiding the need to unfold the model. Thus for a given model description MD and a bound k, the size of the representation is $|MD| + k$, as opposed to $|MD| \times k$ when translating MD to a propositional formula. Experimental results show a significant improvement over a different method of model checking using DL, and comparable performance with BDD-based model checking.

While performance is still not competitive with SAT-based approach, we believe that model checking using DL reasoning is worth exploring. One future direction is to better exploit *absorption* [15]. Absorption is a pre-processing technique that allows the elimination of some forms of concept inclusions by converting them into augmented concept definitions. Our current translation into DL does not allow absorption for most of the concept inclusions.

Acknowledgements

We thank Dmitry Tsarkov for his support in the installation of the $FaCT^{++}$ Description Logic reasoner. We thank Vlad Ciubotariu for his help in running NuSMV.

References

1. NuSMV examples collection. http://nusmv.irst.itc.it/examples/examples.html.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
3. I. Beer, S. Ben-David, and A. Landver. On-the-fly model checking of RCTL formulas. In *Proc. 10^{th} International Conference on Computer Aided Verification (CAV'98)*, LNCS 1427, pages 184–194. Springer-Verlag, 1998.
4. S. Ben-David, R. Trefler, and G. Weddell. Model checking the basic modalities of CTL with description logic. In *Proc. 2006 International Workshop on Description Logics (DL'06)*, pages 223–230.
5. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *TACAS'99*, 1999.
6. R. Bryant. Graph-based algorithms for boolean function manipulation. In *In IEEE Transactions on Computers*, volume c-35 no. 8.

7. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model verifier. In *Computer Aided Verification*, pages 495–499, July 1999.
8. E. Clarke and E. 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.
9. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 2000.
10. N. Dershowitz, Z. Hanna, and J. Katz. Bounded model checking with QBF. In *Eighth International Conference on Theory and Applications of Satisfiability Testing*, pages 408–414, june 2005.
11. A. Dovier and E. Quintarelli. Model checking based data retrieval. In *Revised Papers from the 8th International Workshop on Database Programming Languages, LNCS Vol. 2397*, pages 62–77, 2001.
12. G. Gottlob, E. Grädel, and H. Veith. Linear Time Datalog for Branching Time Logic. In J. Minker, editor, *Logic-Based Artificial Intelligence*, chapter 19. Kluwer, 2000.
13. G. Gottlob, E. Grädel, and H. Veith. Datalog LITE: a deductive query language with linear time model checking. *Computational Logic*, 3(1):42–79, 2002.
14. I. Horrocks. The FaCT system. *Lecture Notes in Computer Science*, 1397:307–312, 1998.
15. I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 285–296, 2000.
16. T. Jussila and A. Biere. Compressing bmc encodings with QBF. In *Fourth International Workshop on Bounded Model Checking*, pages 27–39, August 2006.
17. K. McMillan. Symbolic model checking, 1993.
18. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*, 2001.
19. A. Pnueli. The temporal logic of programs. In *18th IEEE Symposium on Foundation of Computer Science*.
20. J. Quielle and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming*, 1982.
21. M. Sahasrabudhe. SQL-based CTL model checking for telephony feature interactions. In *A Master Thesis, Univesity of Waterloo, Ontario, Canada*, 2004.