Checking Inevitability and Invariance Using Description Logic Technology

Shoham Ben-David[1], Richard Trefler[1], Dmitry Tsarkov[2]
and Grant Weddell[1]

1. David R. Cheriton School of Computing Science
University of Waterloo
2. Department of Computer Science
University of Manchester, UK

Technical Report CS-2008-28

November 17, 2008

Checking Inevitability and Invariance Using Description Logic Technology

Shoham Ben-David[1], Richard Trefler[1], Dmitry Tsarkov[2] and Grant Weddell[1]

1. David R. Cheriton School of Computer Science, University of Waterloo
2. Department of Computer Science, University of Manchester, UK

Abstract. Description Logic is a family of knowledge representation formalisms, mainly used to specify ontologies for information systems. We show how Description Logic can serve as a natural setting for representing and solving symbolic model checking problems. We concentrate on inevitability ($\mathsf{AF}(p)$) specification and invariance ($\mathsf{AG}(p)$) formulas. Experimental results, using the Description Logic reasoner FaCT++, outperform existing methods for inevitability formulas. For invariance formulas we give new encodings that significantly improve on previous implementations using DL technology.

1 Introduction

Symbolic model checking of hardware models is performed using two main methods. The first is based on BDDs and is known as SMV [10], and the second is based on Satisfiability solving technology [4]. We consider a different approach for symbolic model checking, one that makes use of Description Logic (DL) technology. In [3] it was shown how DL can provide a natural setting for Bounded Model Checking (BMC) problems. In this paper we extend the results of [3] in two directions. First, we show how unbounded *inevitability* questions (such as "does event 'p' occur at least once along all computations?") can be naturally phrased and solved, outperforming existing methods. Second, we present two new encodings for BMC safety problems that significantly improve over the results of [3], although still fall behind the performance of SAT solvers for the same task.

Description Logic (c.f. [1]), which can be viewed as a notational variant of modal logic, is a family of knowledge representation formalisms mainly used for specifying ontologies in information systems. Statements in DL are interpreted as decidable fragments of first order logic. The basic elements in DL are *concepts* (sets of individuals) and *roles* (binary relations between individuals).

An ontology $\mathcal{T}$ is called a *terminology* or a TBox, and corresponds to a set of concept inclusions. Each inclusion has the form $\mathtt{C}_1 \sqsubseteq \mathtt{C}_2$, and asserts containment properties of relevant concepts in an underlying domain. For example, we can say that the set of *cows* is included in the set of *animals*

$$\text{COW} \sqsubseteq \text{ANIMAL}$$

and, given a role *eats*, we can assert that the set of *cows* is included in *those things that do not eat animals*

$$\text{COW} \sqsubseteq \forall \textit{eats}.\neg\text{ANIMAL}$$

In addition to concept inclusions, DLs allow for assertions about individuals in the domain. For example, we can say that *Dina* is a cow "COW(*Dina*)" and that *Dina* eats the food *Grass* "*eats*(*Dina*,*Grass*)". A set of assertions is called an ABox.

Checking *knowledge base consistency* is the main reasoning service provided by DL reasoners. For a given terminology $\mathcal{T}$ and set of assertions $\mathcal{A}$ the DL reasoner determines if there exists an interpretation satisfying both the inclusions in $\mathcal{T}$ and the set of assertions in $\mathcal{A}$. Some DL reasoners optimize consistency checking by providing services for answering *concept satisfiability* questions. For a given terminology $\mathcal{T}$ and a concept $\mathtt{C}$, a concept consistency check determines if there exists a non-empty interpretation of the concept $\mathtt{C}$ that also satisfies each inclusion dependency in $\mathcal{T}$. We use $\mathcal{T} \models_{dl} \mathtt{C}$ to denote this fact[1]. Most DL systems implement these services by employing some form of tableaux or model building technique. In recent years several DL reasoners have been developed [8, 7, 13], demonstrating growing ability to solve knowledge base consistency problems.

We cast a model checking problem as a consistency question in DL. 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 $\mathtt{V}_i$, and the transition relation as a single role $\mathtt{R}$. We build a TBox $\mathcal{T}$ by introducing concept inclusions of the type

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

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

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 through 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. Finally, we add to $\mathcal{T}$ a concept inclusion representing a *buggy* path though the model. Verification is then done by checking consistency of the TBox $\mathcal{T}$. Since interpretations of $\mathcal{T}$ correspond to sub-models of M containing a buggy path, if an interpretation is found ($\mathcal{T}$ is consistent) it means that a bug exists, and the interpretation can serve as a counterexample.

The definition of a buggy path depends on the type of specification given. For an inevitability property of the form $\mathsf{AF}(p)$ we define a path along which $\neg p$ always holds, therefore verifying the formula $\mathsf{EG}(\neg p)$. To achieve this we introduce the concepts P and $\mathtt{EGnotP}$ and the inclusion

$$\mathtt{EGnotP} \sqsubseteq \neg \mathsf{P} \sqcap \exists \mathsf{R}.\mathtt{EGnotP}$$

If this inclusion is satisfied, and $\mathtt{EGnotP}$ is not empty, it means that an infinite path (a loop) exists in the model, on which p never holds. For a safety property of the form $\mathsf{AG}(p)$ we define a bounded path on which $\neg p$ appears at least once. We describe two ways to achieve this that are different from the one given in [3].

We present experimental results using the Description Logic reasoner FaCT++ [14]. For inevitability formulas, our results significantly outperform those of BDD based model checking, and are compatible with runs using a SAT solver. It should be noted however, that while SAT solvers perform *bounded* model checking, our method is unbounded, and performs well whether the specification holds in the model or not. For invariance formulas we give new encodings that significantly improve on previous implementations using DL technology, although cannot compete with SAT based BMC.

The rest of the paper is organized as follows. In the next section we give the necessary definitions. Sections 3 and 4 are the main sections of the paper, where we present our encodings for unbounded inevitability specifications (3) and bounded safety specifications (4), prove their correctness and present experimental results. Section 5 concludes the paper.

2 Background and Definitions

We introduce Description Logic in Section 2.1. In section 2.2 we give model checking definitions. Section 2.3 summarizes the results of [3] that we base upon in this paper.

2.1 Description Logic

Description logics usually correspond to decidable fragments of first order logic based on a variable free syntax. The basic elements are *atomic concepts*, which can be understood as unary predicates denoting sets of individuals, and *atomic roles*, which can be understood as binary predicates denoting arbitrary binary relations over individuals. More complicated concepts are constructed from simpler concepts by using so-called *concept constructors*.

A particular description logic is determined by a selection of concept constructors. There is also a naming convention based on this selection, starting with the name $\mathcal{AL}$, short for *attributive language*, which denotes a fragment shared by virtually all description logics. In the following, we define the syntax and semantics for the dialect $\mathcal{ALCOIF}$ which, by virtue of the appended sequence of letters $\mathcal{C}$, $\mathcal{O}$, $\mathcal{I}$ and $\mathcal{F}$, adds concept constructors to $\mathcal{AL}$ to express *concept compliment*, *nominals*, *role inverse* and *role functionality*, respectively.

In the remainder of the paper, we refer to various special cases of this dialect by simply omitting any letters for concept constructors that are not used in particular reductions. For example, $\mathcal{ALCF}$ refers to a dialect which adds concept compliment and functional roles to the basic attributive language.

Definition 1 (Description Logic $\mathcal{ALCOIF}$) *Let* NC, NR *and* NI *be disjoint sets of* atomic concepts $\{A_1, A_2, \ldots\}$, atomic roles $\{R_1, R_2, \ldots\}$ *and* individual names $\{s_1, s_2, \ldots\}$ *respectively. Also let* $\mathsf{NFR} \subset \mathsf{NR}$ *denote a distinguished subset of roles that are* functional. *The set of* concepts C *is the smallest set including* NC *that satisfies the following.*

- *(the basic $\mathcal{AL}$ fragment) If $C_1, C_2 \in \mathsf{C}$ and $R \in \mathsf{NR} - \mathsf{NFR}$, then so are $C_1 \sqcap C_2$ and $\exists R.C$.*
- *(concept compliment $\mathcal{C}$) If $C \in \mathsf{C}$, then so is $\neg C$.*

– *(nominals $\mathcal{O}$) Any finite set $\{s_{i_1}, ..., s_{i_k}\}$ of individual names is in* C.
– *(role inverse $\mathcal{I}$) If $R \in \mathsf{NR} - \mathsf{NFR}$, then $\exists R^-.C \in \mathsf{C}$.*
– *(functional roles $\mathcal{F}$) If $R \in \mathsf{NFR}$, then $\exists R.C \in \mathsf{C}$.*
– *(role inverse and functional roles $\mathcal{IF}$) If $R \in \mathsf{NFR}$, then $\exists R^-.C \in \mathsf{C}$.*

Individual names appearing in concept expressions are called nominals*. A* general concept inclusion *(GCI) is an expression of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are arbitrary concepts. A* terminology *(or* TBox*) $\mathcal{T}$ consists of a finite set of concept inclusions. An* assertion *is an expression of the form $C(s)$, or of the form $R(s_1, s_2)$. An* assertion box *(or* ABox*) $\mathcal{A}$ consists of a finite set of assertions. A* knowledge base *is a pair $(\mathcal{T}, \mathcal{A})$ consisting of a TBox and an ABox.*

The semantics *of expressions is defined with respect to a structure $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, called an* interpretation*, where $\Delta^{\mathcal{I}}$ is a non-empty set of individuals, and $(\cdot)^{\mathcal{I}}$ is an interpretation function that maps atomic concepts A to a subset of $\Delta^{\mathcal{I}}$, atomic roles R to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and individual names s to an element of $\Delta^{\mathcal{I}}$. If $R \in \mathsf{NFR}$, then $R^{\mathcal{I}}$ must also be functional, that is, for any $e, e_1, e_2 \in \Delta^{\mathcal{I}}$, $\{(e, e_1), (e, e_2)\} \subseteq R^{\mathcal{I}}$ implies $e_1 = e_2$. The interpretation function is extended to arbitrary concepts in a way that satisfies each of the following:*

– $(C_1 \sqcap C_2)^{\mathcal{I}} = (C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}$,
– $(\exists R.C)^{\mathcal{I}} = \{e \in \Delta^{\mathcal{I}} : \exists (e, e') \in R^{\mathcal{I}} \text{ s.t. } e' \in C^{\mathcal{I}}\}$,
– $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
– $(\{s_{i_1}, ..., s_{i_k}\})^{\mathcal{I}} = \{(s_1)^{\mathcal{I}}, ..., (s_k)^{\mathcal{I}}\}$, *and*
– $(\exists R^-.C)^{\mathcal{I}} = \{e \in \Delta^{\mathcal{I}} : \exists (e', e) \in R^{\mathcal{I}} \text{ s.t. } e' \in C^{\mathcal{I}}\}$.

An interpretation $\mathcal{I}$ satisfies a GCI $(C_1 \sqsubseteq C_2)$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$, and a TBox *$\mathcal{T}$ if it satisfies each concept inclusion in $\mathcal{T}$. The interpretation satisfies an assertion $C(s)$ if $s^{\mathcal{I}} \in C^{\mathcal{I}}$, an assertion $R(s_1, s_2)$ if $(s_1, s_2) \in R^{\mathcal{I}}$ and an* ABox *$\mathcal{A}$ if it satisfies each assertion in $\mathcal{A}$.*

The knowledge base consistency problem *is to determine, for a given knowledge base $(\mathcal{T}, \mathcal{A})$, if there exists an interpretation $\mathcal{I}$ that satisfies both* TBox *$\mathcal{T}$ and* ABox *$\mathcal{A}$. A variation of this problem is to determine* concept consistency relative to a knowledge base*; that is, to determine, for a given* TBox *$\mathcal{T}$,* ABox *$\mathcal{A}$ and concept C, if $(\mathcal{T}, \mathcal{A} \cup \{C(s)\})$ is consistent, where s is an individual that does not occur in $\mathcal{A}$. This latter problem is written $(\mathcal{T}, \mathcal{A}) \models_{dl} C$, or just $\mathcal{T} \models_{dl} C$ if $\mathcal{A}$ is empty.*

Additional concepts are defined as syntactic sugaring of those above:

$$\begin{aligned}
\text{HERBIVORE} &\sqsubseteq \forall eats.\neg\text{ANIMAL} \\
\text{OMNIVORE} &\sqsubseteq (\exists eats.\text{ANIMAL}) \sqcap \\
&\quad\ (\exists eats.\neg\text{ANIMAL}) \\
\text{COW} &\sqsubseteq \text{ANIMAL} \sqcap \text{HERBIVORE} \\
\text{HUMAN} &\sqsubseteq \text{ANIMAL} \sqcap \\
&\quad\ (\text{HERBIVORE} \sqcup \text{OMNIVORE})
\end{aligned}$$

(a) *The* TBox *(general terminology)*

$$\text{COW}(\text{Dina})$$
$$\text{HUMAN}(\text{Mary})$$
$$likes(\text{Dina}, \text{Mary})$$

(b) *The* ABox *(specific assertions)*

Fig. 1. A KNOWLEDGE BASE ABOUT EATING HABITS.

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

An example knowledge base expressed in $\mathcal{ALCOIF}$ that captures a terminology about animals and their eating habits together with some assertions about specific individuals is given in Figure 1. Our notational convention in this example is to use upper case only for atomic concepts, italic font for atomic roles and a mixed case for individual names. Observe that the TBox introduces, e.g., the concept of an OMNIVORE as *those things that must eat something that is an animal and something else that is not an animal*. Also, the ABox asserts that *there is a cow called Dina that likes Mary, who is a human*.

In this paper, we employ a standard service provided by a tableau-based description logic reasoner, such as FaCT++ [14] for determining knowledge base satisfiability. Such a service will find a compact description of a (possibly infinite) interpretation that satisfies both the TBox and ABox for a given knowledge base if any such interpretation can exist.

2.2 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 in 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 $s \models \neg v_i$.

In practice, the full Kripke structure of a system is not explicitly given. Rather, a model is described by 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* [10].

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 defines 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,

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

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 $v_i = 0$.

Computation Tree Logic (CTL) [6] Given a finite set AP of atomic propositions, formulas of CTL are recursively defined as follows:

– Every atomic proposition is a CTL formula.

– If φ and ψ are CTL formulas then so are:
  • $\neg\varphi$ • $\varphi \wedge \psi$ • $AX\varphi$
  • $EX\varphi$ • $A[\varphi U \psi]$ • $E[\varphi U \psi]$

Additional operators are defined as syntactic sugaring of those above:
• $AF\varphi = A[true\, U \varphi]$ • $EF\varphi = E[true\, U \varphi]$
• $AG\varphi = \neg E[true\, U \neg\varphi]$ • $EG\varphi = \neg A[true\, U \neg\varphi]$

The formal semantics of a CTL formula are defined with respect to a Kripke structure $M = (S, I, R, L)$ over a set of variables $V = \{v_1, ..., v_k\}$. A path in M is an infinite sequence of states $(s_0, s_1, ...)$ such that each successive pair of states (s_i, s_{i+1}) is an element of R. The notation $M, s \models \varphi$, means that the formula φ is true in state s of the model M.

– $M, s \models p$ iff $s \models p$
– $M, s \models \neg\varphi$ iff $M, s \not\models \varphi$
– $M, s \models \varphi \wedge \psi$ iff $M, s \models \varphi$ and $M, s \models \psi$
– $M, s_0 \models\ AXp$ iff for all paths $(s_0, s_1, ...)$, $M, s_1 \models p$
– $M, s_0 \models\ EXp$ iff exists a path $(s_0, s_1, ...)$, $M, s_1 \models p$
– $M, s_0 \models A[\varphi U \psi]$ iff for all paths $(s_0, s_1, ...)$, there exists i such that $M, s_i \models \psi$ and for all $j < i$,$M, s_j \models \varphi$
– $M, s_0 \models E[\varphi U \psi]$ iff there exists a path $(s_0, s_1, ...)$, and there exists i such that $M, s_i \models \psi$ and for all $j < i$,$M, s_j \models \varphi$

We say that a Kripke structure $M = (S, I, R, L)$ satisfies a CTL formula φ ($M \models \varphi$) if there exists a state s_i such that $s_i \models I$ and $M, s_i \models \varphi$.

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 $\varphi = \mathsf{AG}(p)$, we say that $M^k \not\models \varphi$ if and only if there exists a path $w = s_0, ..., s_j$ in M such that $j \leq k$ and $s_j \models \neg p$.

2.3 Model description as a DL terminology

We show how a model description can be written as a TBox over the Description Logic dialect $\mathcal{ALC}$. This translation is taken from [3]. Using this translation for model checking is the main contribution of this paper, and will be shown in Sections 3 and 4.

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 $V = \{v_1, ..., v_n\}$.

We generate a TBox $\mathcal{T}_{MD}$, linear in the size of MD. 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 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 $\sqcap, \sqcup, \neg$ respectively. The concepts $\mathsf{C}_i, \mathsf{C}'_i$ correspond to the Boolean conditions c_i, c'_i in the same way. We then introduce concept inclusions describing the model: for each pair $\langle c_i, c'_i\rangle$ we introduce the inclusions

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

For a model description MD over n variables ($V = \{v_1, ..., v_n\}$), $\mathcal{T}_{MD}$ will consists of $2n$ concept inclusions. Interpretations for $\mathcal{T}_{MD}$ correspond to sub-models of M_{MD}. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for $\mathcal{T}_{MD}$. The correlation between individuals from $\Delta^{\mathcal{I}}$ and states in S can be seen by defining a function $F : \Delta^{\mathcal{I}} \to S$ 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 F is well defined, since a state s is determined by the value of the variables $v_1, ..., v_n$.

The following Lemma (taken from [3]) extends the correlation to Boolean expressions over $v_1, ..., v_n$.

Lemma 4. *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation for $\mathcal{T}_{MD}$. 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 $\sigma \in \mathsf{C}^{\mathcal{I}}$ if and only if $s \models c$.*

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

Corollary 5. *Let M_{MD}, $\mathcal{T}_{MD}$, $\mathcal{I}$ and F be as above. Let $F(\sigma_1) = s_1$ and $F(\sigma_2) = s_2$. Then $(s_1, s_2) \in R$ if and only if $(\sigma_1, \sigma_2) \in \mathsf{R}^{\mathcal{I}}$.*

Proof. (sketch).
$(s_1, s_2) \in R$ if and only if s_1, s_2 obey the definition regarding the Boolean expression pairs $\langle c_i, c'_i\rangle$ for $0 < i \leq n$. By Lemma 4, this happens if and only if σ_1, σ_2 behave similarly regarding the concepts $\mathsf{C}_i, \mathsf{C}'_i$, which happens if and only if $(\sigma_1, \sigma_2) \in \mathsf{R}^{\mathcal{I}}$. □

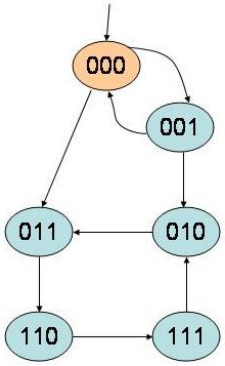

Fig. 2. A Kripke structure for Counter

We demonstrate the construction of $\mathcal{T}_{MD}$ with the following example.

Example 6 *Consider a model of a buggy three-bit counter, for which the least and most significant bits behave as expected, but the middle bit has a bug: when its current value is 0, it may assume any value in the next state, and when its current value is 1 it keeps its value in the next state. This behavior can be described as a model description (using T for $v_1 \vee \neg v_1$ and F for $v_1 \wedge \neg v_1$) in the following way.*
$\texttt{Counter} =$

$$(I, [\langle v_1, T\rangle, \langle F, v_2\rangle, \langle (v_1 \wedge v_2 \wedge v_3) \vee (\neg v_3 \wedge (\neg v_1 \vee \neg v_2)), T\rangle])$$

with $I = \neg v_1 \wedge \neg v_2 \wedge \neg v_3$. Figure 2 describes the Kripke structure for $\texttt{Counter}$.

The description of the model as a TBox $\mathcal{T}_{\texttt{Counter}}$ *over $\mathcal{ALC}$ has three concepts $\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3$ and one role R. The concept inclusions for $\mathcal{T}_{\texttt{Counter}}$ are given in Figure 2.1. For convenience we broke the concept inclusions describing the behavior of V_3 into two parts. Note that there is only one concept inclusion describing the behavior of V_2, since it is free to change when its value is 0.*

3 Inevitability Formulas using DL

In this section we consider formulas of the type $\mathsf{AF}(p)$ with p being a Boolean expression. For our method we need to define a buggy path, that

$$
\begin{array}{rcl}
\mathsf{S}_0 & \sqsubseteq & (\neg \mathsf{V}_1 \sqcap \neg \mathsf{V}_2 \sqcap \neg \mathsf{V}_3) \\
\mathsf{V}_1 & \sqsubseteq & \forall \mathsf{R}.\neg \mathsf{V}_1 \\
\neg \mathsf{V}_1 & \sqsubseteq & \forall \mathsf{R}.\mathsf{V}_1 \\
\mathsf{V}_2 & \sqsubseteq & \forall \mathsf{R}.\mathsf{V}_2 \\
(\mathsf{V}_1 \sqcap \mathsf{V}_2 \sqcap \mathsf{V}_3) & \sqsubseteq & \forall \mathsf{R}.\neg \mathsf{V}_3 \\
(\neg \mathsf{V}_3 \sqcap (\neg \mathsf{V}_1 \sqcup \neg \mathsf{V}_2)) & \sqsubseteq & \forall \mathsf{R}.\neg \mathsf{V}_3 \\
(\mathsf{V}_1 \sqcap \mathsf{V}_2 \sqcap \neg \mathsf{V}_3) & \sqsubseteq & \forall \mathsf{R}.\mathsf{V}_3 \\
(\mathsf{V}_3 \sqcap (\neg \mathsf{V}_1 \sqcup \neg \mathsf{V}_2)) & \sqsubseteq & \forall \mathsf{R}.\mathsf{V}_3
\end{array}
$$

Fig. 3. The TBox $\mathcal{T}_{Counter}$

is, a path on which p never holds. We thus look for a representation of $\mathsf{EG}(\neg p)$.

A CTL formula can be identified with the set of states in which it holds [6]. Looking at it this way, we get the following equation.

$$\mathsf{EG}(\neg p) = \neg p \wedge \mathsf{EX}(\mathsf{EG}(\neg p)) \tag{1}$$

We use this equation for our translation into DL. 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 $V = \{v_1, ..., v_n\}$, and let $\mathcal{T}_{MD}$ be the terminology built for it as describe in Section 2.3. Let $\varphi = \mathsf{AF}(p)$ be the formula to be verified, with p being a Boolean expression over the variables $v_1, ..., v_n$. Let P be the concept representing the Boolean expression p, by replacing every state variable v_i with the concept V_i, and $\wedge, \vee, \neg$ with $\sqcap, \sqcup, \neg$ respectively.

We introduce a new concept called `EGnotP`, and add the following concept inclusion to $\mathcal{T}_{MD}$:

$$\mathtt{EGnotP} \sqsubseteq \neg \mathsf{P} \sqcap \exists \mathsf{R}.\mathtt{EGnotP} \tag{2}$$

Note that the expression $\exists \mathsf{R}.\mathsf{C}$ can be seen as taking one step through R, and thus corresponds, in a sense, to the CTL expression $\mathsf{EX}(\mathsf{C})$.

Let $\mathcal{T}'_{MD}$ be the terminology we get by adding Equation (2) to $\mathcal{T}_{MD}$. We define the concept $\mathsf{C}_\varphi \equiv \mathsf{S}_0 \sqcap \mathtt{EGnotP}$. In order to verify φ, we now check whether C_φ is consistent with respect to our terminology: $\mathcal{T}'_{MD} \models_{dl} \mathsf{C}_\varphi$?

A positive answer from the DL reasoning tool will be accompanied by an interpretation for $\mathcal{T}'_{MD}$ in which C_φ is not empty. This interpretation can serve an a witness to $\mathsf{EG}(\neg p)$, or as a counterexample to $\mathsf{AF}(p)$.

The following proposition states our result formally.

Proposition 7. *$M_{MD} \not\models \varphi$ if and only if $\mathcal{T}'_{MD} \models_{dl} \mathsf{C}_\varphi$.*

Proof. ($\Longrightarrow$). Assume that $M_{MD} \not\models \varphi$. Since M_{MD} is a finite kripke structure, this means there exists a loop, that is, a sequence of states s_0, s_1, ... ,s_m such that $s_0 \models I$, $s_i \not\models p$ for $0 \leq i \leq m$, $(s_i, s_{i+1}) \in R$ for $0 \leq i < m$, and $s_m = s_j$ for some $0 \leq j < m$. We use this sequence to build an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ for $\mathcal{T}'_{MD}$. We define m individuals $\sigma_0, \sigma_1, ..., \sigma_{m-1}$ in $\Delta^\mathcal{I}$, that correspond to s_0, s_1, ... ,s_{m-1}. We map $\sigma_i \in \mathtt{EGnotP}^\mathcal{I}$ for $0 \leq i < m$. We then map each σ_i to the primitive concepts $\mathsf{V}_k^\mathcal{I}$ according to s_i as expected: $\sigma_i \in \mathsf{V}_k^\mathcal{I}$ if and only if $s_i \models v_k$. Note that since $s_0 \models I$ we get by Lemma 4 that $\sigma_0 \in \mathsf{S}_0^\mathcal{I}$, and also $\sigma_i \notin \mathsf{P}^\mathcal{I}$ for $0 \leq i < m$, since $s_i \not\models p$. We define $(\sigma_i, \sigma_{i+1}) \in \mathsf{R}^\mathcal{I}$ for $0 \leq i < m-1$, and also $(\sigma_{m-1}, \sigma_j) \in \mathsf{R}^\mathcal{I}$. It is easy to see that the interpretation $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}'_{MD}$:

- 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 a 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 4 we get that $\sigma_{l-1} \in \mathsf{C}_i^\mathcal{I}$ implies $\sigma_l \notin \mathsf{V}_i^\mathcal{I}$ and $\sigma_{l-1} \in (\Delta^\mathcal{I} \setminus \mathsf{C}_i^\mathcal{I}) \cap \mathsf{C}_i'^\mathcal{I}$ implies $\sigma_l \in \mathsf{V}_i^\mathcal{I}$. Since no pairs other than (σ_{l-1}, σ_l) and (σ_{m-1}, σ_j) belong to $\mathsf{R}^\mathcal{I}$ in the interpretation $\mathcal{I}$, the inclusions hold.
- The inclusion $\mathtt{EGnotP} \sqsubseteq \neg \mathsf{P} \sqcap \exists \mathsf{R}.\mathtt{EGnotP}$. By the construction of $\mathcal{I}$, all individuals σ_i belong to $\mathtt{EGnotP}^\mathcal{I}$. We know also that $\sigma_i \notin \mathsf{P}^\mathcal{I}$. Since each individual has an outgoing edge that is also in $\mathtt{EGnotP}^\mathcal{I}$ the inclusion holds.
- C_φ is not empty since $\sigma_0 \in \mathsf{S}_0^\mathcal{I} \cap \mathtt{EGnotP}^\mathcal{I}$.

($\Longleftarrow$). Assume that $\mathcal{T}'_{MD} \models_{dl} \mathsf{C}_\varphi$. Then there exists an interpretation for $\mathcal{T}'_{MD}$, such that $\mathsf{C}_\varphi^\mathcal{I}$ is not empty. Since $\mathcal{ALC}$ enjoys the *finite model property* [1], there must exist a finite interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ for $\mathcal{T}'_{MD}$ such that $\mathsf{C}_\varphi^\mathcal{I}$ is not empty. Thus there exists an individual $\sigma_0 \in \mathsf{S}_0^\mathcal{I} \cap \mathtt{EGnotP}^\mathcal{I}$.

Since $\sigma_0 \in \mathtt{EGnotP}^\mathcal{I}$ and $\mathtt{EGnotP}^\mathcal{I} \subseteq (\Delta^\mathcal{I} \setminus \mathsf{P}^\mathcal{I}) \cap \{e \in \Delta^\mathcal{I} : \exists (e, e') \in \mathsf{R}^\mathcal{I}$ s.t. $e' \in \mathtt{EGnotP}^\mathcal{I}\}$ we know that $\sigma_0 \notin \mathsf{P}^\mathcal{I}$, and there must exist $\sigma_1 \in \mathtt{EGnotP}^\mathcal{I}$ such that $(\sigma_0, \sigma_1) \in \mathsf{R}^\mathcal{I}$. For similar considerations, there exists a sequence of individuals, $\sigma_0, \sigma_1, \sigma_2, ...$, such that $\sigma_i \in \mathtt{EGnotP}^\mathcal{I}$, $\sigma_i \notin \mathsf{P}^\mathcal{I}$, and $(\sigma_i, \sigma_{i+1}) \in \mathsf{R}^\mathcal{I}$ for all i. Since $\mathcal{I}$ is finite

there must exist m, j such that $\sigma_m = \sigma_j$. We show that $M_{MD} \not\models \varphi$ by presenting an infinite sequence of states (a loop) in M_{MD} that do not satisfy p. We map each σ_i to a state s_i as usual: $\sigma_i \in \mathsf{V}_k^{\mathcal{I}}$ if and only if $s_i \models v_k$. By Lemma 4, $s_i \not\models p$, since $\sigma_i \notin \mathsf{P}^{\mathcal{I}}$. Also, by Corollary 5, $\forall 0 \leq i \leq m$, $(s_i, s_{i+1}) \in R$ and $(s_m, s_j) \in R$.

□

It is tempting to try and use the same reasoning to verify a formula $\psi = \mathsf{AG}(p)$: instead of the concept inclusion in (2), add the concept AGp and the following concept inclusion:

$$\mathsf{AGp} \sqsubseteq \mathsf{P} \sqcap \forall \mathsf{R}.\mathsf{AGp}. \tag{3}$$

Define $\mathsf{C}_\psi \equiv \mathsf{S}_0 \sqcap \mathsf{AGp}$. Let $\mathcal{T}''_{MD}$ be the terminology we get by replacing Equation (2) with Equation (3) in $\mathcal{T}'_{MD}$. Note that checking $\mathcal{T}''_{MD} \models_{dl} \mathsf{C}_\psi$ does not give us what we want. To see this, recall that $\mathcal{T}''_{MD} \models_{dl} \mathsf{C}_\psi$ asks whether *there exists* an interpretation $\mathcal{I}$, that satisfies all concept inclusions in $\mathcal{T}''_{MD}$, and for which C_ψ is not empty. Such interpretation does not necessarily include all possible transitions in the given model M_{MD}. In fact, an interpretation that satisfies inclusion (2) would be enough for inclusion (3) as well. Thus $\mathcal{T}''_{MD} \models_{dl} \mathsf{C}_\psi$ verifies $\mathsf{EG}(p)$ and not $\mathsf{AG}(p)$.

For $\mathsf{AG}(p)$ formulas, we can only achieve bounded model checking. Before we show this (Section 4), we discuss experimental results for inevitability formulas.

3.1 Experimental results

We demonstrate the usefulness of this method for $\mathsf{AF}(p)$ formulas in Table I. We ran our encoding using the description logic reasoner *FACT++* [14]. We used a model derived from the NuSMV example "dme1-16", taken from [12]. We parameterized this example, to be able to run models with different number of variables. We give results for three model sizes, consisting of 85, 170 and 272 state variables, and for two formula types: one that holds in the model (*passes*) and one that *fails* to hold in the model.

We compare our method both to BDD based models checking and to SAT based method that performs bounded model checking. We note that while translation of safety properties into a CNF formula for SAT is supported by publicly available model checking tools, such as NuSMV [5] and Cadence-SMV [9], it is not the case for liveness formulas like $\mathsf{AF}(p)$.

Model Size	Formula	DL	BDD	SAT
85	Fail	0.05	52	0.48
85	Pass	0.05	124	∞
170	Fail	0.1	> 1200	1
170	Pass	0.1	> 1200	∞
272	Fail	0.2	> 1200	1.5
272	Pass	0.17	> 1200	∞

Table 1. $\mathsf{AF}(p)$ formulas runtime

In fact, the only model checker we found to support $\mathsf{AF}(p)$ formulas in BMC mode was RuleBase [2].

The time in Table I is given in seconds. We have set a limit of 20 minutes for each run. Note that while results using SAT solving are close to those using DL for formulas that *fail* in the model, there is no guaranteed termination for formulas that *pass*.

4 BMC of safety formulas using DL

Bounded model checking of invariance formulas using Description Logic was first presented in [3]. The method in [3] used the DL dialect $\mathcal{ALCI}$ (that allows for *Inverse Roles*), and was based on taking backward steps through R from concepts representing states of certain distance from the initial state (also known as *doughnuts*). We present here two alternative encodings, using ABoxes, that significantly improve over the one given in [3]. While these encodings still fail to compete with SAT based BMC, we find the improvement demonstrated by the new methods to be encouraging evidence supporting further research in this direction.

Let MD be a model description, $\varphi = \mathsf{AG}(p)$ an invariance formula to be verified, and k the bound. Our encodings consist of three parts. The first translates MD into a TBox $\mathcal{T}_{MD}$. This part is described in Section 2.3, and is not repeated here. The second part encodes a symbolic path of length k as an ABox $\mathcal{A}_k$. This part is given in Section 4.1 below. The third part of our encodings deals with the verification of the given formula φ. We present two different encodings. The first makes use of *nominals* while the second requires *functional roles*. Both encodings are described in section 4.2. In section 4.3 we demonstrate these translations with an example, and experimental results comparing the two methods

to the one of [3] and to SAT based BMC are given in Section 4.4. We discuss the methods and results in Section 4.5.

4.1 Constructing $\mathcal{A}_k$.

For a bound k, we introduce $k+1$ individuals, $\mathsf{s}_0, \mathsf{s}_1, ..., \mathsf{s}_k$. We then introduce one concept assertion: $\mathsf{S}_0(\mathsf{s}_0)$ and k role assertions: $\forall 0 \leq i < k$, $\mathsf{R}(\mathsf{s}_i, \mathsf{s}_{i+1})$.
Note that the assertions in $\mathcal{A}_k$ form a symbolic path of length $k+1$ through the model, starting from an initial state. Note also that this syntactic definition of a path does not depend on the model described in the TBox $\mathcal{T}_{MD}$ and would be the same for any model and formula.

4.2 Encoding the formula

Let $\varphi = \mathsf{AG}(p)$ be the specification to be verified, with p being a Boolean formula over the variables $v_1, ..., v_n$. Note that for any model M, $M \models \mathsf{AG}(p)$ if and only if $M \not\models \mathsf{EF}(\neg p)$. We use this fact in our encodings. We translate the Boolean formula p into a concept P in the usual way, and then encode a possible *bug* in the model, thus verifying $\mathsf{EF}(\neg p)$. We present two ways to achieve this.

Using Nominals Define the concept $\mathsf{C}_\varphi \equiv \neg\mathsf{P} \sqcap \{\mathsf{s}_0, ..., \mathsf{s}_k\}$. If C_φ is consistent with respect to $(\mathcal{T}_{MD}, \mathcal{A}_k)$, it means that $\neg p$ holds in some state, with distance k or less from the initial state. Verification is therefore reduced to the query: $(\mathcal{T}_{MD}, \mathcal{A}_k) \models_{dl} \mathsf{C}_\varphi$.

Using Functional Roles This encoding is based on Clarke and Emerson's [6] fixpoint representation of $\mathsf{EF}(p)$:

$$\mathsf{EF}(p) = p \vee \mathsf{EX}(\mathsf{EF}(p))$$

In order to encode this in DL, we need to enhance both the TBox $\mathcal{T}_{MD}$ and the ABox $\mathcal{A}_k$.

We first define R to be a *functional role*, to ensure that each individual in the interpretation has at most one outgoing edge through R. We then add an assertion to $\mathcal{A}_k$:

$$(\neg\exists\mathsf{R}.\top)(\mathsf{s}_k).$$

Note that individuals belonging to the concept $\neg\exists\mathsf{R}.\top$ have no outgoing edges. The assertion above thus forces $\mathtt{s}_k$ to be the last state in the path.

We then build the TBox $\mathcal{T}'_{MD}$ by adding one concept inclusion to $\mathcal{T}_{MD}$. We introduce a new concept $\mathtt{EFnotP}$, and define it as follows:

$$\mathtt{EFnotP} \sqsubseteq \neg\mathsf{P} \sqcup \exists\mathsf{R}.\mathtt{EFnotP}$$

The intuition behind this concept inclusion is the following. We first check whether $\neg\mathsf{P}$ holds in the current state; if it does, then a bug was found and we are done. If not, we try to perform the same check on the following states, that are accessible via the role R. Since R is a functional, we have that $\exists\mathsf{R}.\mathtt{EFnotP}$ is the same as $\forall\mathsf{R}.\mathtt{EFnotP}$, and it is propagated to the next state. If $\neg\mathsf{P}$ does not hold in the last state, $\exists\mathsf{R}.\mathtt{EFnotP}$ is not applicable anymore, and the search stops after k steps.

Finally, we add another assertion to $\mathcal{A}_k$, stating that $\mathtt{s}_0$, the initial state, belongs also to the new concept $\mathtt{EFnotP}$:

$$\mathtt{EFnotP}(\mathtt{s}_0).$$

Let $\mathcal{A}'_k = \mathcal{A}_k \cup \{\mathtt{EFnotP}(\mathtt{s}_0), \neg\exists\mathsf{R}.\top(\mathtt{s}_k)\}$, and $\mathcal{T}'_{MD}$ as defined above. If $(\mathcal{T}'_{MD}, \mathcal{A}'_k)$ is consistent, it means that $\neg p$ holds on one of the states of distance k or less from the initial state.

The following proposition states that both our encodings are correct.

Proposition 8.

1. $M^k_{MD} \not\models \varphi$ *if and only if* $(\mathcal{T}_{MD}, \mathcal{A}_k) \models_{dl} \mathsf{C}_\varphi$.
2. $M^k_{MD} \not\models \varphi$ *if and only if* $(\mathcal{T}'_{MD}, \mathcal{A}'_k)$ *is consistent.*

We show only one direction of the proof. The other direction follows easily by similar arguments.

Proof. ($\Longrightarrow$) 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 $s_0 \models I$, $\forall 0 < i \leq j, (s_{i-1}, s_i) \in R$, and $s_j \not\models p$. We build a finite interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ based on w. The set $\Delta^\mathcal{I}$ will include $k+1$ elements $\sigma_0, ..., \sigma_k$. The first $j+1$ of them, $\sigma_0, ..., \sigma_j$, will correspond to $s_0, ..., s_j$ in a way that $\forall 0 < l \leq n$, $\forall 0 \leq i \leq j$, $\sigma_i \in \mathsf{V}^\mathcal{I}_l$ if and only if $s_i \models v_l$. We define $(\sigma_i, \sigma_{i+1}) \in \mathsf{R}^\mathcal{I}$ for $0 \leq i \leq k$, and interpret the individuals $\mathtt{s}_0, ..., \mathtt{s}_k$ of $\mathcal{A}_k$ as $\sigma_0, ..., \sigma_k$.

Since $s_j \not\models p$, we get by Lemma 4 that $\sigma_j \notin \mathsf{P}^{\mathcal{I}}$. Also, the concept inclusions of $\mathcal{T}_{MD}$ hold (see proof of Proposition 7).

We have to show that $\mathcal{I}$ satisfies both translations.

1. $(\mathcal{T}_{MD}, \mathcal{A}_k) \models_{dl} \mathsf{C}_\varphi$. By the construction of $\mathcal{I}$, it satisfies both $\mathcal{T}_{MD}$ and $\mathcal{A}_k$. It remains to be shown that $\mathsf{C}_\varphi^{\mathcal{I}}$ is not empty. Since s_j is interpreted as σ_j and $\sigma_j \notin \mathsf{P}^{\mathcal{I}}$, we get that $\sigma_j \in (\Delta^{\mathcal{I}} \setminus \mathsf{P}^{\mathcal{I}}) \cap \{\mathsf{s}_0^{\mathcal{I}}, ..., \mathsf{s}_k^{\mathcal{I}}\}$ = $\mathsf{C}_\varphi^{\mathcal{I}}$.
2. $(\mathcal{T}'_{MD}, \mathcal{A}'_k)$ is consistent. To show this, we map σ_i, $0 \leq i \leq j$, to belong to $\mathtt{EFnotP}$. The assertions in $\mathcal{A}'_k$ therefore hold: $(\mathsf{s}_i^{\mathcal{I}}, \mathsf{s}_{i+1}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$, $\mathsf{s}_0^{\mathcal{I}} \in \mathtt{EFnotP}^{\mathcal{I}}$ and $\mathsf{s}_k^{\mathcal{I}} \in (\Delta^{\mathcal{I}} \setminus \{e \in \Delta^{\mathcal{I}} : \exists (e, e') \in \mathsf{R}^{\mathcal{I}}\})$ since there is no outgoing edge from $\mathsf{s}_k^{\mathcal{I}}$. It remains to be shown that the inclusion $\mathtt{EFnotP} \sqsubseteq \neg \mathsf{P} \sqcup \exists \mathsf{R}.\mathtt{EFnotP}$ holds under the interpretation $\mathcal{I}$. We know that $\sigma_0, \sigma_1, ..., \sigma_j \in \mathtt{EFnotP}^{\mathcal{I}}$, and only them. We have to show that these individuals belong also to the right hand side of the inclusion. $\sigma_0, \sigma_1, ..., \sigma_{j-1}$ belong there since $(\sigma_i, \sigma_{i+1}) \in \mathsf{R}^{\mathcal{I}}$, $\sigma_{i+1} \in \mathtt{EFnotP}^{\mathcal{I}}$ for $0 \leq i < j$. Since $\sigma_j \notin \mathsf{P}^{\mathcal{I}}$ it also belongs there.

□

4.3 Example

Consider the Kripke structure in Example 6, Section 2.3. Let the formula to be verified be $\varphi = AG\neg(v_1 \wedge \neg v_2 \wedge \neg v_3)$, asserting that the state $(1, 0, 0)$ is not reachable. Let the bound be set to 4. Let $\mathcal{T}_{\mathtt{Counter}}$ be as in Example 6. The ABox $\mathcal{A}_4$ representing a path of length 4 in the model, consists of the following assertions:

$$\mathcal{A}_4 = \{\mathsf{S}_0(\mathsf{s}_0), \mathsf{R}(\mathsf{s}_0, \mathsf{s}_1), \mathsf{R}(\mathsf{s}_1, \mathsf{s}_2), \mathsf{R}(\mathsf{s}_2, \mathsf{s}_3), \mathsf{R}(\mathsf{s}_3, \mathsf{s}_4)\}$$

We now have two ways to verify the formula.

1. Using nominals, we define

   $$\mathsf{C}_\varphi \equiv (\mathsf{V}_1 \sqcap \neg \mathsf{V}_2 \sqcap \neg \mathsf{V}_3) \sqcap \{\mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_3, \mathsf{s}_4\}.$$

   Verification is then carried out by checking the satisfiability of C_φ: $(\mathcal{T}_{\mathtt{Counter}}, \mathcal{A}_4) \models_{dl} \mathsf{C}_\varphi$?
2. Using functional roles, we define R to be a functional role, and add the following inclusion to create $\mathcal{T}'_{\mathtt{Counter}}$:

   $$\mathtt{EFnotP} \sqsubseteq (\mathsf{V}_1 \sqcap \neg \mathsf{V}_2 \sqcap \neg \mathsf{V}_3) \sqcup \exists \mathsf{R}.\mathtt{EFnotP}$$

We then add two assertions to $\mathcal{A}_4$:

$$\mathcal{A}'_4 = \mathcal{A}_4 \cup \{\mathtt{EFnotP}(\mathtt{s}_0), \neg\exists \mathtt{R}.\top(\mathtt{s}_4)\}$$

Verification is now carried out by asking whether $(\mathcal{T}'_{\mathtt{Counter}}, \mathcal{A}'_4)$ is consistent.

Note that in both cases we expect the DL reasoner to give an "unsatisfiable" or "inconsistent" result, since the formula φ holds in `Counter`.

4.4 Experimental Results

We implemented the methods and ran experiments using the description logic reasoner *FACT++* [14]. We used the same model as described in Section 3.1, but ran different model sizes, consisting of 85, 272 and 425 state variables. We present results of running several different bounds for each model. Table 2 presents results comparing the runtime (in seconds) of four methods. We set a time limit of 1200 seconds. We use $--$ to denote runs exceeding this limit. The column titled $\mathcal{ALCI}$ represents the method of [3], using inverse roles, and the columns titled $\mathcal{ALCO}$ and $\mathcal{ALCF}$ give results of running the methods presented in this paper. The last column gives results of running similar models with a SAT solver. We used the BMC mode of Cadence-SMV [9], that invoked *zChaff* [11] as a SAT solver for bounded model checking.

As can be seen in Table 2, both the methods described in this paper outperform the results of [3], with the method based on a functional role performing significantly better than the one using nominals. As the bound increases, however, the SAT solver performs much better than all DL methods.

4.5 Discussion

The improvement in performance achieved when revising the encoding to use ABox assertions instead of inverse roles, is not surprising. In the inverse roles encoding, individuals were dynamically created. Since this process is highly non-deterministic, the reasoner had to recreate the nodes again and again, affecting the performance negatively. When the nodes of the model are fixed, as in the case of a path described by ABox assertions, the reasoner needs only to find a valuation of the boolean variables in each node.

Model Size	Bound	$\mathcal{ALCI}$	$\mathcal{ALCO}$	$\mathcal{ALCF}$	SAT
85	5	1.77	0.01	0	0.51
85	6	287	0.02	0	0.63
85	7	−−	0.02	0.01	0.68
85	9	−−	0.04	0.03	0.9
85	11	−−	−−	0.19	1.04
85	13	−−	−−	1.44	1.23
85	15	−−	−−	5.29	1.37
85	17	−−	−−	80.37	1.52
85	20	−−	−−	−−	2.11
272	5	14	0.16	0.12	1.23
272	6	−−	0.19	0.13	1.68
272	7	−−	0.22	0.14	2.36
272	9	−−	0.35	0.23	3.45
272	11	−−	−−	1.17	3.61
272	13	−−	−−	7.32	4.50
272	15	−−	−−	26.21	4.78
272	17	−−	−−	−−	5.34
272	20	−−	−−	−−	6.03
425	5	32.48	0.75	0.67	0.71
425	6	−−	0.80	0.65	0.72
425	7	−−	0.82	0.69	0.78
425	9	−−	3.43	0.86	2.36
425	11	−−	−−	3.03	3.44
425	13	−−	−−	13.99	3.80
425	15	−−	−−	41.75	4.24
425	17	−−	−−	−−	5.67
425	20	−−	−−	−−	6.72

Table 2. BMC runtime using DL and SAT

Using the nominal approach, we again cause the reasoner to perform redundant work. In this method, the reasoner has to guess a "violation point" (that is, choose a nominal for which the bug should be found), then build an interpretation and try to satisfy all the restrictions given in the TBox. In this process, some values may need to be propagated through the whole interpretation, requiring a significant amount of effort from the reasoner. When a clash is found, all of this work is thrown away, and needs to be repeated when a new nominal is guessed as a "violation point". In contrast, when using functionals, we let the reasoner work "linearly", fixing the value in a point before all other branching decisions are made. This allows the system to find a clash earlier in the process, and move to the next step without doing as much redundant work.

5 Conclusion

We have presented several new approaches for using DL reasoning technology to solve model checking problems. DL allows for natural encodings of model descriptions that are significantly more concise than encoding used in SAT- solvers to solve bounded model checking problems.

For specifications of unbounded inevitability properties our method uses a DL reasoner to check for restrictions of the model that satisfy formulas of the form $\mathsf{EG}(p)$. We have shown experimental evidence that this method outperforms BDD based tools, has a performance cost that is in line with SAT-based reasoning tools, and yet, in contrast with SAT-techniques, is not restricted to answering bounded model checking questions.

When the specifications are invariance properties, we have provided two new bounded model checking methods, that significantly outperform earlier approaches using DL reasoners. In particular, the method that uses a logic with functional roles offers significant improvement.

For the future, we plan to perform more case studies on a variety of models and specifications to better understand when the use of DL reasoning technology can provide a significant advantage.

Acknowledgements

This work was partially supported by the SEALIFE project (IST-2006-027269), by NSERC of Canada and by the Cheriton Scholarship Fund of the Cheriton School of Computer Science.

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, C. Eisner, and A. Landver. RuleBase: An industry-oriented formal verification tool. In *In Proc. 34^{th} Design Automation Conference (DAC96)*, pages 655–660, 1996.
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*, LNAI 4548-0060, pages 60–72, July 2007.
4. A. Biere, A. Cimatti, E. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *In Proc. fifth international TOOLS AND ALGORITHMS FOR THE CONSTRUCTION AND ANALYSIS OF SYSTEMS (TACAS)*, 1999.

5. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model verifier. In *Computer Aided Verification*, pages 495–499, July 1999.
6. 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.
7. GmbH and Co. KG. *Racer Systems*. RacerPro Software, http://www.racer-systems.com, 2005.
8. I. Horrocks. The FaCT system. *Lecture Notes in Computer Science*, 1397:307–312, 1998.
9. K. McMillan. Cadence-smv. http://www.kenmcmil.com/smv.html.
10. K. McMillan. Symbolic model checking, 1993.
11. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *39th Design Automation Conference*, 2001.
12. NuSMV examples collection. http://nusmv.irst.itc.it/examples/examples.html.
13. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2), 2007.
14. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.