# Description Logic Reasoning with Decision Diagrams

## Compiling $\mathcal{SHIQ}$ to Disjunctive Datalog*

Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler

Institut AIFB, Universität Karlsruhe, Germany
{sru,mak,phi}@aifb.uni-karlsruhe.de

**Abstract.** We propose a novel method for reasoning in the description logic $\mathcal{SHIQ}$. After a satisfiability preserving transformation from $\mathcal{SHIQ}$ to the description logic $\mathcal{ALCI}b$, the obtained $\mathcal{ALCI}b$ Tbox $\mathcal{T}$ is converted into an ordered binary decision diagram (OBDD) which represents a canonical model for $\mathcal{T}$. This OBDD is turned into a disjunctive datalog program that can be used for Abox reasoning. The algorithm is worst-case optimal w.r.t. data complexity, and admits easy extensions with DL-safe rules and ground conjunctive queries.

## 1 Introduction

In order to leverage intelligent applications for the Semantic Web, scalable reasoning systems for the standardised Web Ontology Language OWL[1] are required. OWL is essentially based on description logics (DLs), with the DL known as $\mathcal{SHIQ}$ currently being among its most prominent fragments. State-of-the art OWL reasoners, such as Pellet, FaCT++, or RacerPro use tableau methods with good performance results, but even those successful systems are not applicable in all practical cases. This motivates the search for alternative reasoning approaches that build upon different methods in order to address cases where tableau algorithms turn out to have certain weaknesses. Successful examples are recent works based on resolution and hyper-tableau calculi, as realised by the systems KAON2 and HermiT.

In this paper, we pursue a new DL reasoning paradigm based on the use of ordered binary decision diagrams (OBDD). These reasoning tools have been successfully applied in the domain of large-scale model checking and verification, but have hitherto seen only little investigation in DLs [1]. Our work bases on a recent adoption of OBDDs for terminological reasoning in $\mathcal{SHIQ}$ [2]. This approach, however, is inherently inapt of dealing with assertional knowledge directly. We therefore adopt the existing OBDD method for terminological reasoning, but use its output for generating a disjunctive datalog program that can in turn be combined with Abox data to obtain a correct reasoning procedure. The main technical contribution of the paper is to show this adoption to be sound and complete based on suitable model constructions. Considering possible applications, the work establishes the basis for applying OBDD-based methods for $\mathcal{SHIQ}$ reasoning, including natural support for DL-safe rules and ground queries.

* Supported by the European Commission under contracts 027595 NeOn and 215040 ACTIVE, and by the Deutsche Forschungsgemeinschaft (DFG) under the ReaSem project.

[1] http://www.w3.org/2004/OWL/

A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 435–450, 2008.

The structure of the paper is as follows. In Section 2, we recall some essential definitions and results on which we base our approach. Section 3 then discusses the decomposition of models into sets of *dominoes*, which are then computed with OBDDs in Section 4. The resulting OBDD presentation is transformed to disjunctive datalog in Section 5, where we also show the correctness of the approach. Section 6 concludes.

## 2 The Description Logics $\mathcal{SHIQ}$ and $\mathcal{ALCIb}$

We first recall some basic definitions of DLs (see [3] for a comprehensive treatment of DLs) and introduce our notation. Next we define a rather expressive description logic $\mathcal{SHIQ}b$ that extends $\mathcal{SHIQ}$ with restricted Boolean role expressions [4]. We will not consider $\mathcal{SHIQ}b$ knowledge bases, but the DL serves as a convenient umbrella logic for the DLs used in this paper.

**Definition 1.** *A $\mathcal{SHIQ}b$ knowledge base is based on three disjoint sets of* concept names $\mathsf{N}_C$*,* role names $\mathsf{N}_R$*, and* individual names $\mathsf{N}_I$*. A set of* atomic roles $\mathbf{R}$ *is defined as* $\mathbf{R} := \mathsf{N}_R \cup \{R^- \mid R \in \mathsf{N}_R\}$*. In addition, we set* $\mathrm{Inv}(R) := R^-$ *and* $\mathrm{Inv}(R^-) := R$*, and we will extend this notation also to sets of atomic roles. In the sequel, we will use the symbols $R, S$ to denote atomic roles, if not specified otherwise.*

*The set of* Boolean role expressions $\mathbf{B}$ *is defined as*

$$\mathbf{B} ::= \mathbf{R} \mid \neg\mathbf{B} \mid \mathbf{B} \sqcap \mathbf{B} \mid \mathbf{B} \sqcup \mathbf{B}.$$

*We use $\vdash$ to denote standard Boolean entailment between sets of atomic roles and role expressions. Given a set $\mathcal{R}$ of atomic roles, we inductively define:*

- *For atomic roles $R$, $\mathcal{R} \vdash R$ if $R \in \mathcal{R}$, and $\mathcal{R} \nvdash R$ otherwise,*
- *$\mathcal{R} \vdash \neg U$ if $\mathcal{R} \nvdash U$, and $\mathcal{R} \nvdash \neg U$ otherwise,*
- *$\mathcal{R} \vdash U \sqcap V$ if $\mathcal{R} \vdash U$ and $\mathcal{R} \vdash V$, and $\mathcal{R} \nvdash U \sqcap V$ otherwise,*
- *$\mathcal{R} \vdash U \sqcup V$ if $\mathcal{R} \vdash U$ or $\mathcal{R} \vdash V$, and $\mathcal{R} \nvdash U \sqcup V$ otherwise.*

*A Boolean role expression $U$ is* restricted *if $\emptyset \nvdash U$. The set of all restricted role expressions is denoted* $\mathbf{T}$*, and the symbols $U$ and $V$ will be used throughout this paper to denote restricted role expressions. A $\mathcal{SHIQ}b$* Rbox *is a set of axioms of the form $U \sqsubseteq V$ (role inclusion axiom) or* $\mathsf{Tra}(R)$ *(transitivity axiom). The set of non-simple roles (for a given Rbox) is inductively defined as follows:*

- *If there is an axiom* $\mathsf{Tra}(R)$*, then $R$ is non-simple.*
- *If there is an axiom $R \sqsubseteq S$ with $R$ non-simple, then $S$ is non-simple.*
- *If $R$ is non-simple, then* $\mathrm{Inv}(R)$ *is non-simple.*

*A role is* simple *if it is atomic (simplicity of Boolean role expressions is not relevant in this paper) and not non-simple. Based on a $\mathcal{SHIQ}b$ Rbox, the set of* concept expressions $\mathbf{C}$ *is the smallest set containing* $\mathsf{N}_C$*, and all concept expressions given in Table 1, where $C, D \in \mathbf{C}$, $U \in \mathbf{T}$, and $R \in \mathbf{R}$ is a simple role. Throughout this paper, the symbols $C$, $D$ will be used to denote concept expressions. A $\mathcal{SHIQ}b$* Tbox *(or* terminology*) is a set of* general concept inclusion axioms *(GCIs) of the form $C \sqsubseteq D$. A $\mathcal{SHIQ}b$* Abox *(containing* assertional knowledge*) is a set of statements of the form $C(a)$ or $R(a, b)$, where $a, b \in \mathsf{N}_I$. We assume throughout that all roles and concepts occurring in the*

**Table 1.** Semantics of constructors in $\mathcal{SHIQ}b$ for an interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}}$

| Name | Syntax | Semantics |
|---|---|---|
| inverse role | $R^-$ | $\{\langle x,y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$ |
| role negation | $\neg U$ | $\{\langle x,y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle x,y\rangle \notin U^{\mathcal{I}}\}$ |
| role conj. | $U \sqcap V$ | $U^{\mathcal{I}} \cap V^{\mathcal{I}}$ |
| role disj. | $U \sqcup V$ | $U^{\mathcal{I}} \cup V^{\mathcal{I}}$ |
| top | $\top$ | $\Delta^{\mathcal{I}}$ |
| bottom | $\bot$ | $\emptyset$ |
| negation | $\neg C$ | $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| disjunction | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| univ. rest. | $\forall U.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in U^{\mathcal{I}}$ implies $y \in C^{\mathcal{I}}\}$ |
| exist. rest. | $\exists U.C$ | $\{x \in \Delta^{\mathcal{I}} \mid y \in \Delta^{\mathcal{I}} \colon \langle x,y\rangle \in U^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$ |
| qualified | $\leq n\, R.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \leq n\}$ |
| number rest. | $\geq n\, R.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid \langle x,y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}$ |

*Abox are atomic (which can be done without loss of generality). A* $\mathcal{SHIQ}b$ knowledge base KB *is a triple* $\langle \mathcal{A}, \mathcal{R}, \mathcal{T}\rangle$*, where* $\mathcal{A}$ *is an Abox,* $\mathcal{R}$ *is an Rbox, and* $\mathcal{T}$ *is a Tbox.*

As mentioned above, we will consider only fragments of $\mathcal{SHIQ}b$. In particular, a $\mathcal{SHIQ}$ knowledge base is a $\mathcal{SHIQ}b$ knowledge base without Boolean role expressions, and an $\mathcal{ALCI}b$ knowledge base is a $\mathcal{SHIQ}b$ knowledge base that contains no Rbox axioms and no number restrictions (i.e. axioms $\leq n\, R.C$ or $\geq n\, R.C$). Consequently, an $\mathcal{ALCI}b$ knowledge base only consists of a pair $\langle \mathcal{A}, \mathcal{T}\rangle$, where $\mathcal{A}$ is an Abox and $\mathcal{T}$ is a Tbox. The related DL $\mathcal{ALCQI}b$ has been studied in [4].

An interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called *domain* (the elements of it being called *individuals*) together with a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of $\Delta^{\mathcal{I}}$, concept names to subsets of $\Delta^{\mathcal{I}}$, and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is extended to role and concept expressions as shown in Table 1. An interpretation $\mathcal{I}$ *satisfies* an axiom $\varphi$ if we find that $\mathcal{I} \models \varphi$, where

- $\mathcal{I} \models U \sqsubseteq V$ if $U^{\mathcal{I}} \subseteq V^{\mathcal{I}}$,
- $\mathcal{I} \models \mathsf{Tra}(R)$ if $R^{\mathcal{I}}$ is a transitive relation,
- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,
- $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
- $\mathcal{I} \models R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

$\mathcal{I}$ *satisfies* a knowledge base KB, $\mathcal{I} \models$ KB, if it satisfies all axioms of KB. *Satisfiability*, *equivalence*, and *equisatisfiability* of knowledge bases are defined as usual.

For convenience of notation, we abbreviate Tbox axioms of the form $\top \sqsubseteq C$ by writing just $C$. Statements such as $\mathcal{I} \models C$ and $C \in$ KB are interpreted accordingly. Note that $C \sqsubseteq D$ can thus be written as $\neg C \sqcup D$.

Finally, we will often need to access a particular set of quantified and atomic subformulae of a DL concept. These specific parts are provided by the function $P : \mathbf{C} \to 2^{\mathbf{C}}$:

$$P(C) := \begin{cases} P(D) & \text{if } C = \neg D \\ P(D) \cup P(E) & \text{if } C = D \sqcap E \text{or } C = D \sqcup E \\ \{C\} \cup P(D) & \text{if } C = \mathsf{Q}U.D \text{ with } \mathsf{Q} \in \{\exists, \forall, \geq n, \leq n\} \\ \{C\} & \text{otherwise} \end{cases}$$

We generalise $P$ to DL knowledge bases KB by defining $P(\mathrm{KB})$ to be the union of the sets $P(C)$ for all Tbox axioms $C$ in KB.

We will usually express all Tbox axioms as simple concept expressions as explained above. Given a knowledge base KB we obtain its negation normal form $\mathsf{NNF}(\mathrm{KB})$ by converting every Tbox concept into its negation normal form as usual. It is well-known that KB and $\mathsf{NNF}(\mathrm{KB})$ are equivalent.

For $\mathcal{ALCI}b$ knowledge bases KB, we will usually require another normalisation step that simplifies the structure of KB by *flattening* it to a knowledge base $\mathsf{FLAT}(\mathrm{KB})$. This is achieved by transforming KB into negation normal form and exhaustively applying the following transformation rules:

- Select an outermost occurrence of $\mathsf{Q}U.D$ in KB, such that $\mathsf{Q} \in \{\exists, \forall\}$ and $D$ is a non-atomic concept.
- Substitute this occurrence with $\mathsf{Q}U.F$ where $F$ is a fresh concept name (i.e. one not occurring in the knowledge base).
- Add $\neg F \sqcup D$ to the knowledge base.

Obviously, this procedure terminates yielding a flat knowledge base $\mathsf{FLAT}(\mathrm{KB})$ all Tbox axioms of which are Boolean expressions over formulae of the form $A$, $\neg A$, or $\mathsf{Q}U.A$ with $A$ an atomic concept name. As shown in [2], any $\mathcal{ALCI}b$ knowledge base KB is equisatisfiable to $\mathsf{FLAT}(\mathrm{KB})$. This work also detailed a reduction of $\mathcal{SHIQ}$ knowledge bases to $\mathcal{ALCI}b$ that we summarise as follows:

**Theorem 2.** *Any $\mathcal{SHIQ}$ knowledge base* KB *can be transformed in polynomial time into an equisatisfiable $\mathcal{ALCI}b$ knowledge base* KB′.

It is easy to see that the algorithm from [2] is still applicable in the presence of Aboxes, and that ground Abox conclusions are preserved – with the exception of entailments of the form $R(a, b)$ for non-simple roles $R$ which fall victim to the standard elimination of transitivity axioms.

## 3 Building Models from Domino Sets

Our approach towards terminological reasoning in $\mathcal{ALCI}b$ exploits the fact that models for this DL can be decomposed into small parts, which we call *dominoes*. Intuitively, each domino abstractly represents two individuals in an $\mathcal{ALCI}b$ interpretation, based on their concept properties and role relationships. We will see that suitable sets of such two-element pieces suffice to reconstruct models of $\mathcal{ALCI}b$ Tboxes, and satisfiability of $\mathcal{ALCI}b$ terminologies can thus be reduced to the existence of suitable sets.

We first introduce the basic notion of a domino set, and its relationship to interpretations. Given a DL language with concepts $\mathbf{C}$ and roles $\mathbf{R}$, a *domino* is an arbitrary triple $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$, where $\mathcal{A}, \mathcal{B} \subseteq \mathbf{C}$ and $\mathcal{R} \subseteq \mathbf{R}$. We will generally assume a fixed language and refer to dominoes over that language only. Interpretations can be deconstructed into sets of dominoes as follows:

**Definition 3.** *Given an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, and a set $C \subseteq \mathbf{C}$ of concept expressions, the* domino projection *of $\mathcal{I}$ w.r.t. $C$, denoted by $\pi_C(\mathcal{I})$ is the set that contains for all $\delta, \delta' \in \Delta^{\mathcal{I}}$ the triple $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ with*

– $\mathcal{A} = \{C \in \mathbf{C} \mid \delta \in C^{\mathcal{I}}\}$,
– $\mathcal{R} = \{R \in \mathbf{R} \mid \langle \delta, \delta' \rangle \in R^{\mathcal{I}}\}$,
– $\mathcal{B} = \{C \in \mathbf{C} \mid \delta' \in C^{\mathcal{I}}\}$.

An inverse construction of interpretations from arbitrary domino sets is as follows:

**Definition 4.** *Given a set $\mathbb{D}$ of dominoes, the induced* domino interpretation $\mathcal{I}(\mathbb{D}) = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *is defined as follows:*

1. *$\Delta^{\mathcal{I}}$ consists of all finite nonempty words over $\mathbb{D}$ where, for each pair of subsequent letters $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ and $\langle \mathcal{A}', \mathcal{R}', \mathcal{B}' \rangle$ in a word, we have $\mathcal{B} = \mathcal{A}'$.*
2. *For $\delta = \langle \mathcal{A}_1, \mathcal{R}_1, \mathcal{A}_2 \rangle \langle \mathcal{A}_2, \mathcal{R}_2, \mathcal{A}_3 \rangle \ldots \langle \mathcal{A}_{i-1}, \mathcal{R}_{i-1}, \mathcal{A}_i \rangle$ a word and $A \in \mathsf{N}_C$ a concept name, we define* $\mathrm{tail}(\delta) := \mathcal{A}_i$*, and set $\delta \in A^{\mathcal{I}}$ iff $A \in \mathrm{tail}(\delta)$,*
3. *For each $R \in \mathsf{N}_R$, we set $\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}}$ if either $\delta_2 = \delta_1 \langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ with $R \in \mathcal{R}$ or $\delta_1 = \delta_2 \langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ with $\mathrm{Inv}(R) \in \mathcal{R}$.*

Mark that – following the intuition – the domino interpretation is constructed by conjoining matching dominoes. This process is also similar to the related method of "unravelling" models in order to obtain tree-like interpretations.

Domino projections do not faithfully represent the structure of the interpretation that they were constructed from, yet they capture enough information to reconstruct models of a Tbox $\mathcal{T}$, as long as $\mathbf{C}$ is chosen to contain at least $P(\mathcal{T})$. Indeed, it was shown in [2] that, for any $\mathcal{ALCIb}$ terminology $\mathcal{T}$, $\mathcal{J} \models \mathcal{T}$ iff $\mathcal{I}(\pi_{P(\mathcal{T})}(\mathcal{J})) \models \mathcal{T}$. This observation allows us to devise an algorithm that directly constructs a suitable domino set from which one could obtain a model that witnesses the satisfiability of some knowledge base. The following algorithm therefore considers all possible dominoes, and iteratively eliminates those that cannot occur in the domino projection of any model:

**Definition 5.** *Consider an $\mathcal{ALCIb}$ terminology $\mathcal{T}$, and define $\mathbf{C} = P(\mathsf{FLAT}(\mathcal{T}))$. Sets $\mathbb{D}_i$ of dominoes based on concepts from $\mathbf{C}$ are constructed as follows:*

*$\mathbb{D}_0$ consists of all dominoes $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ which satisfy:*

**kb:** *for every concept $C \in \mathsf{FLAT}(\mathcal{T})$, we have that $\bigsqcap_{D \in \mathcal{A}} D \sqsubseteq C$ is a tautology*[2]*,*
**ex:** *for all $\exists U.A \in \mathbf{C}$, if $A \in \mathcal{B}$ and $\mathcal{R} \vdash U$ then $\exists U.A \in \mathcal{A}$,*
**uni:** *for all $\forall U.A \in \mathbf{C}$, if $\forall U.A \in \mathcal{A}$ and $\mathcal{R} \vdash U$ then $A \in \mathcal{B}$.*

*Given a domino set $\mathbb{D}_i$, the set $\mathbb{D}_{i+1}$ consists of all dominoes $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle \in \mathbb{D}_i$ satisfying the following conditions:*

**delex:** *for every $\exists U.A \in \mathbf{C}$ with $\exists U.A \in \mathcal{A}$, there is some $\langle \mathcal{A}, \mathcal{R}', \mathcal{B}' \rangle \in \mathbb{D}_i$ such that $\mathcal{R}' \vdash U$ and $A \in \mathcal{B}'$,*
**deluni:** *for every $\forall U.A \in \mathbf{C}$ with $\forall U.A \notin \mathcal{A}$, there is some $\langle \mathcal{A}, \mathcal{R}', \mathcal{B}' \rangle \in \mathbb{D}_i$ such that $\mathcal{R}' \vdash U$ but $A \notin \mathcal{B}'$,*
**sym:** $\langle \mathcal{B}, \mathrm{Inv}(\mathcal{R}), \mathcal{A} \rangle \in \mathbb{D}_i$.

*The construction of domino sets $\mathbb{D}_{i+1}$ is continued until $\mathbb{D}_{i+1} = \mathbb{D}_i$. The final result $\mathbb{D}_{\mathcal{T}} := \mathbb{D}_{i+1}$ defines the* canonical domino set *of $\mathcal{T}$.*

[2] Note that formulae in $\mathsf{FLAT}(\mathcal{T})$ and in $\mathcal{A} \subseteq \mathbf{C}$ are such that this can easily be checked by evaluating the Boolean operators in $\mathbf{C}$ as if $\mathcal{A}$ was a set of true propositional variables.

Note that the algorithm must terminate, since it starts from a finite initial set $\mathbb{D}_0$ that is reduced in each computation step. Intuitively, the algorithm implements a kind of greatest fixed point construction that yields the domino projection of the largest possible model of the terminological part of an $\mathcal{ALCIb}$ knowledge base. The following result makes this intuition more explicitly:

**Lemma 6.** *Consider an $\mathcal{ALCIb}$ terminology $\mathcal{T}$ and an arbitrary model $\mathcal{I}$ of $\mathcal{T}$. Then the domino projection $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I})$ is contained in $\mathbb{D}_\mathcal{T}$.*

*Proof.* The claim is shown by a simple induction. In the following, we use $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ to denote an arbitrary domino of $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I})$. For the base case, we must show that $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I}) \subseteq \mathbb{D}_0$. Let $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ to denote an arbitrary domino of $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I})$ which was generated from elements $\langle \delta, \delta' \rangle$. Then $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ satisfies condition **kb**, since $\delta \in C^\mathcal{I}$ for any $C \in \mathsf{FLAT}(\mathcal{T})$. The conditions **ex** and **uni** are obviously satisfied.

For the induction step, assume that $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I}) \subseteq \mathbb{D}_i$, and let $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ again denote an arbitrary domino of $\pi_{P(\mathsf{FLAT}(\mathcal{T}))}(\mathcal{I})$ which was generated from elements $\langle \delta, \delta' \rangle$.

- For **delex**, note that $\exists U.A \in \mathcal{A}$ implies $\delta \in (\exists U.A)^\mathcal{I}$. Thus there is an individual $\delta''$ such that $\langle \delta, \delta'' \rangle \in U^\mathcal{I}$ and $\delta'' \in A^\mathcal{I}$. Clearly, the domino generated by $\langle \delta, \delta'' \rangle$ satisfies the conditions of **delex**.
- For **deluni**, note that $\forall U.A \notin \mathcal{A}$ implies $\delta \notin (\forall U.A)^\mathcal{I}$. Thus there is an individual $\delta''$ such that $\langle \delta, \delta'' \rangle \in U^\mathcal{I}$ and $\delta'' \notin A^\mathcal{I}$. Clearly, the domino generated by $\langle \delta, \delta'' \rangle$ satisfies the conditions of **deluni**.
- The condition of **sym** for $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ is clearly satisfied by the domino generated from $\langle \delta', \delta \rangle$. □

We will also exploit this observation in the later construction of models for knowledge bases with individual assertions. The following was again shown in [2]:

**Theorem 7.** *An $\mathcal{ALCIb}$ terminology $\mathcal{T}$ is satisfiable iff its canonical domino set $\mathbb{D}_\mathcal{T}$ is non-empty. Definition 5 thus defines a decision procedure for satisfiability of $\mathcal{ALCIb}$ terminologies.*

## 4 Sets as Boolean Functions

The algorithm of the previous section may seem to be of little practical use, since it requires the computations on an exponentially large set of dominoes. The required computation steps, however, can also be accomplished with a more indirect representation of the possible dominoes based on Boolean functions. Indeed, any propositional logic formula represents a set of interpretations for which the function evaluates to *true*. Using a suitable encoding, each interpretation can be understood as a domino, and a propositional formula can represent a domino set.

In order for this approach to be more feasible than the naive algorithm given above, an efficient representation of propositional formulae is needed. For this we use binary decision diagrams (BDDs), that have been applied to represent complex Boolean functions in model-checking (see, e.g., [5]). A particular optimisation of these structures are ordered BDDs (OBDDs) that use a dynamic precedence order of propositional variables to obtain compressed representations. We provide a first introduction to OBDDs below. A more detailed exposition and pointers to the literature are given in [6].

**Boolean Functions and Operations.** We first explain how sets can be represented by means of Boolean functions. This will enable us, given a fixed finite base set $S$, to represent every family of sets $\mathbb{S} \subseteq 2^S$ by a single Boolean function.

A *Boolean function* on a set $\mathsf{Var}$ of variables is a function $\varphi : 2^{\mathsf{Var}} \to \{\mathit{true}, \mathit{false}\}$. The underlying intuition is that $\varphi(V)$ computes the truth value of a Boolean formula based on the assumption that exactly the variables of $V$ are evaluated to *true*. A simple example are so-called *characteristic functions* of the form $[\![v]\!]_\chi$ for some $v \in \mathsf{Var}$, which are defined as $[\![v]\!]_\chi(V) := \mathit{true}$ iff $v \in V$, or the functions $[\![\mathit{true}]\!]$ and $[\![\mathit{false}]\!]$ mapping any input to *true* or *false*, respectively.

Boolean functions over the same set of variables can be combined and modified in several ways. Firstly, there are the obvious Boolean operators for negation, conjunction, disjunction, and implication. By slight abuse of notation, we will use the common (syntactic) operator symbols $\neg$, $\wedge$, $\vee$, and $\to$ to also represent such (semantic) operators on Boolean functions. Given, e.g., Boolean functions $\varphi$ and $\psi$, we find that $(\varphi \wedge \psi)(V) = \mathit{true}$ iff $\varphi(V) = \mathit{true}$ and $\psi(V) = \mathit{true}$. Note that the result of the application of $\wedge$ results in another Boolean function, and is not to be understood as a syntactic formula. Another operation on Boolean functions is existential quantification over a set of variables $V \subseteq \mathsf{Var}$, written as $\exists V.\varphi$ for some function $\varphi$. Given an input set $W \subseteq \mathsf{Var}$ of variables, we define $(\exists V.\varphi)(W) = \mathit{true}$ iff *there is some* $V' \subseteq V$ such that $\varphi(V' \cup (W \setminus V)) = \mathit{true}$. In other words, there must be a way to set truth values of variables in $V$ such that $\varphi$ evaluates to *true*. Universal quantification is defined analogously, and we thus have $\forall V.\varphi := \neg\exists V.\neg\varphi$ as usual. Mark that our use of $\exists$ and $\forall$ overloads notation, and should not be confused with role restrictions in DL expressions.

**Ordered Binary Decision Diagrams.** Binary Decision Diagrams (BDDs), intuitively, are a generalisation of decision trees which allow the reuse of nodes. Structurally, BDDs are directed acyclic graphs whose nodes are labelled by variables from some set $\mathsf{Var}$. The only exception are two *terminal* nodes that are labelled by *true* and *false*, respectively. Every non-terminal node has two outgoing edges, corresponding to the two possible truth values of the variable.

**Definition 8.** *A* BDD *is a tuple* $\mathbb{O} = (N, n_{\mathsf{root}}, n_{\mathsf{true}}, n_{\mathsf{false}}, \mathsf{low}, \mathsf{high}, \mathsf{Var}, \lambda)$ *where*

- $N$ *is a finite set called* nodes*,*
- $n_{\mathsf{root}} \in N$ *is called the* root node*,*
- $n_{\mathsf{true}}, n_{\mathsf{false}} \in N$ *are called the* terminal nodes*,*
- $\mathsf{low}, \mathsf{high} : N \setminus \{n_{\mathsf{true}}, n_{\mathsf{false}}\} \to N$ *are two* child functions *assigning to every non-terminal node a* low *and a* high *child node. Furthermore the graph obtained by iterated application has to be acyclic, i.e. for no node n exists a sequence of applications of* low *and* high *resulting in n again.*
- $\mathsf{Var}$ *is a finite set of* variables.
- $\lambda : N \setminus \{n_{\mathsf{true}}, n_{\mathsf{false}}\} \to \mathsf{Var}$ *is the labelling function assigning to every non-terminal node a variable from* $\mathsf{Var}$.

OBBDs are a particular realisation of BDDs where a certain ordering is imposed on variables to achieve more efficient representations. We will not require to consider the background of this optimisation in here. Now every BDD based on a variable set $\mathsf{Var} = \{x_1, \ldots, x_n\}$ represents an $n$-ary Boolean function $\varphi : 2^{\mathsf{Var}} \to \{\mathit{true}, \mathit{false}\}$.

**Definition 9.** *Given a BDD* $\mathbb{O} = (N, n_{\mathsf{root}}, n_{\mathsf{true}}, n_{\mathsf{false}}, \mathsf{low}, \mathsf{high}, \mathsf{Var}, \lambda)$ *the Boolean function* $\varphi_{\mathbb{O}} : 2^{\mathsf{Var}} \to \{true, false\}$ *is defined recursively as follows:*

$$\varphi_{\mathbb{O}} := \varphi_{n_{\mathsf{root}}} \qquad \varphi_{n_{\mathsf{true}}} = [\![true]\!] \qquad \varphi_{n_{\mathsf{false}}} = [\![false]\!]$$

$$\varphi_n = \left(\neg[\![\lambda(n)]\!]_{\mathcal{X}} \wedge \varphi_{\mathsf{low}(n)}\right) \vee \left([\![\lambda(n)]\!]_{\mathcal{X}} \wedge \varphi_{\mathsf{high}(n)}\right) \textit{for } n \in N \setminus \{n_{\mathsf{true}}, n_{\mathsf{false}}\}$$

In other words, the value $\varphi(V)$ for some $V \subseteq \mathsf{Var}$ is determined by traversing the BDD, beginning from the root node: at a node labelled with $v \in \mathsf{Var}$, the evaluation proceeds with the node connected by the high-edge if $v \in V$, and with the node connected by the low-edge otherwise. If a terminal node is reached, its label is returned as a result.

BDDs for some Boolean formula might be exponentially large in general, but often there is a representation which allows for BDDs of manageable size. Finding the optimal representation is NP-complete, but heuristics have shown to yield good approximate solutions. Hence (O)BDDs are often conceived as efficiently compressed representations of Boolean functions. In addition, many operations on Boolean functions – such as the aforementioned "point-wise" negation, conjunction, disjunction, implication as well as propositional quantification – can be performed directly on the corresponding OBDDs by fast algorithms.

**Translating Dominos into Boolean Functions.** To apply the above machinery to DL reasoning, consider a flattened $\mathcal{ALCIb}$ terminology $\mathcal{T} = \mathsf{FLAT}(\mathcal{T})$. A set of propositional variables Var is defined as $\mathsf{Var} := \mathbf{R} \cup (P(\mathcal{T}) \times \{1, 2\})$. We thus obtain an obvious bijection between sets $V \subseteq \mathsf{Var}$ and dominoes over the set $P(\mathcal{T})$ given as $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle \mapsto (\mathcal{A} \times \{1\}) \cup \mathcal{R} \cup (\mathcal{B} \times \{2\})$. Hence, any Boolean function over Var represents a domino set as the collection of all variable sets for which it evaluates to *true*. We can use this observation to rephrase the construction of $\mathbb{D}_{\mathcal{T}}$ in Definition 5 into an equivalent construction of a function $[\![\mathcal{T}]\!]$.

We first represent DL concepts $C$ and role expressions $U$ by characteristic Boolean functions over Var as follows. Note that the application of $\wedge$ results in another Boolean function, and is not to be understood as a syntactic formula.

$$[\![C]\!] := \begin{cases} \neg[\![D]\!] & \text{if } C = \neg D \\ [\![D]\!] \wedge [\![E]\!] & \text{if } C = D \sqcap E \\ [\![D]\!] \vee [\![E]\!] & \text{if } C = D \sqcup E \\ [\![\langle C, 1\rangle]\!]_{\mathcal{X}} & \text{if } C \in P(\mathcal{T}) \end{cases} \qquad [\![U]\!] := \begin{cases} \neg[\![V]\!] & \text{if } U = \neg V \\ [\![V]\!] \wedge [\![W]\!] & \text{if } U = V \sqcap W \\ [\![V]\!] \vee [\![W]\!] & \text{if } U = V \sqcup W \\ [\![U]\!]_{\mathcal{X}} & \text{if } U \in \mathbf{R} \end{cases}$$

We can now define an inferencing algorithm based on Boolean functions.

**Definition 10.** *Given a flattened* $\mathcal{ALCIb}$ *terminology* $\mathcal{T}$ *and a variable set* Var *defined as above, Boolean functions* $[\![\mathcal{T}]\!]_i$ *are constructed based on the definitions in Fig. 1:*

- $[\![\mathcal{T}]\!]_0 := \varphi^{\mathbf{kb}} \wedge \varphi^{\mathbf{uni}} \wedge \varphi^{\mathbf{ex}}$,
- $[\![\mathcal{T}]\!]_{i+1} := [\![\mathcal{T}]\!]_i \wedge \varphi_i^{\mathbf{delex}} \wedge \varphi_i^{\mathbf{deluni}} \wedge \varphi_i^{\mathbf{sym}}$

*The construction terminates as soon as* $[\![\mathcal{T}]\!]_{i+1} = [\![\mathcal{T}]\!]_i$*, and the result of the construction is then defined as* $[\![\mathcal{T}]\!] := [\![\mathcal{T}]\!]_i$*. The algorithm returns "unsatisfiable" if* $[\![\mathcal{T}]\!](V) = false$ *for all* $V \subseteq \mathsf{Var}$*, and "satisfiable" otherwise.*

As shown in [2], the above algorithm is a correct procedure for checking consistency of terminological $\mathcal{ALCIb}$ knowledge bases. Moreover, all required operations and checks

$$\varphi^{\mathbf{kb}} := \bigwedge_{C\in\mathcal{T}} [\![C]\!]$$

$$\varphi^{\mathbf{uni}} := \bigwedge_{\forall U.C\in P(\mathcal{T})} [\![\langle\forall U.C,1\rangle]\!]_\chi \wedge [\![U]\!] \to [\![\langle C,2\rangle]\!]_\chi \qquad \varphi^{\mathbf{ex}} := \bigwedge_{\exists U.C\in P(\mathcal{T})} [\![\langle C,2\rangle]\!]_\chi \wedge [\![U]\!] \to [\![\langle\exists U.C,1\rangle]\!]_\chi$$

$$\varphi_i^{\mathbf{delex}} := \bigwedge_{\exists U.C\in P(\mathcal{T})} [\![\langle\exists U.C,1\rangle]\!]_\chi \to \exists(\mathbf{R}\cup C\times\{2\}).([\![\mathcal{T}]\!]_i \wedge [\![U]\!] \wedge [\![\langle C,2\rangle]\!]_\chi)$$

$$\varphi_i^{\mathbf{deluni}} := \bigwedge_{\forall U.C\in P(\mathcal{T})} [\![\langle\forall U.C,1\rangle]\!]_\chi \to \neg\exists(\mathbf{R}\cup C\times\{2\}).([\![\mathcal{T}]\!]_i \wedge [\![U]\!] \wedge \neg[\![\langle C,2\rangle]\!]_\chi)$$

$$\varphi_i^{\mathbf{sym}}(V) := [\![\mathcal{T}]\!]_i\Big(\{\langle D,1\rangle \mid \langle D,2\rangle \in V\} \cup \{\mathrm{Inv}(R) \mid R \in V\} \cup \{\langle D,2\rangle \mid \langle D,1\rangle \in V\}\Big)$$

**Fig. 1.** Boolean functions for defining the canonical domino set in Definition 10

$$\begin{aligned} \mathit{PhDStudent} &\sqsubseteq \exists \mathit{has}.\mathit{Diploma} \\ \mathit{Diploma} &\sqsubseteq \forall \mathit{has}^-.\mathit{Graduate} \\ \mathit{Diploma} \sqcap \mathit{Graduate} &\sqsubseteq \top \\ \mathit{Diploma}(\mathit{laureus}) &\quad \mathit{PhDStudent}(\mathit{laureus}) \end{aligned}$$

**Fig. 2.** An example $\mathcal{ALCI}b$ knowledge base

are provided by standard OBDD implementations, and thus can be realised in practice. Correctness follows from the next observation, which is also relevant for extending reasoning to Aboxes below:

**Proposition 11.** *For any $\mathcal{ALCI}b$ terminology $\mathcal{T}$ and variable set $V \in \mathsf{Var}$ as above, we find that $[\![\mathcal{T}]\!](V) = true$ iff $V$ represents a domino in $\mathbb{D}_\mathcal{T}$ as defined in Definition 5.*

In the remainder of this section, we illustrate the above algorithm by an extended example, to which we will also come back to explain the later extensions of the inference algorithm. Therefore, consider the $\mathcal{ALCI}b$ knowledge base given in Fig. 2. For now, we are only interested in the terminological axioms, the consistency of which we would like to establish. As a first transformation step, all Tbox axioms are transformed into the following universally valid concepts in negation normal form:

$$\neg \mathit{PhDStudent} \sqcup \exists \mathit{has}.\mathit{Diploma} \qquad \neg \mathit{Diploma} \sqcup \forall \mathit{has}^-.\mathit{Graduate} \qquad \neg \mathit{Diploma} \sqcup \neg \mathit{Graduate}$$

The flattening step can be skipped since all concepts are already flat. Now the relevant concept expressions for describing dominoes are as follows given by the set $P(\mathcal{T}) = \{\exists \mathit{has}.\mathit{Diploma}, \forall \mathit{has}^-.\mathit{Graduate}, \mathit{Diploma}, \mathit{Graduate}, \mathit{PhDStudent}\}$. We thus obtain the following set $\mathsf{Var}$ of Boolean variables (though $\mathsf{Var}$ is just a set, our presentation follows the domino intuition):

| | | |
|---|---|---|
| $\langle\exists \mathit{has}.\mathit{Diploma},1\rangle$ | $\mathit{has}$ | $\langle\exists \mathit{has}.\mathit{Diploma},2\rangle$ |
| $\langle\forall \mathit{has}^-.\mathit{Graduate},1\rangle$ | $\mathit{has}^-$ | $\langle\forall \mathit{has}^-.\mathit{Graduate},2\rangle$ |
| $\langle \mathit{Diploma},1\rangle$ | | $\langle \mathit{Diploma},2\rangle$ |
| $\langle \mathit{Graduate},1\rangle$ | | $\langle \mathit{Graduate},2\rangle$ |
| $\langle \mathit{PhDStudent},1\rangle$ | | $\langle \mathit{PhDStudent},2\rangle$ |

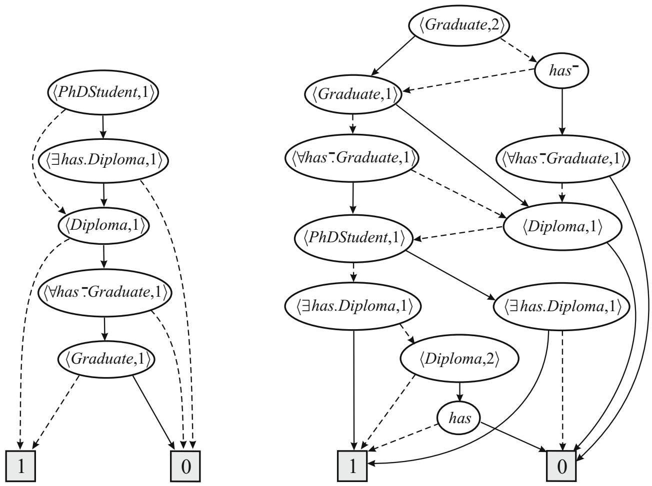


**Fig. 3.** OBDDs arising when processing the terminology of Fig. 2. Following traditional BDD notation, solid arrows indicate high successors, and dashed arrows indicate low successors.

We are now ready to construct the OBDDs as described. Figure 3 (left) displays an OBDD corresponding to the following Boolean function:

$$\begin{aligned}\varphi^{\mathbf{kb}} := & (\neg[\![\langle \mathit{PhDStudent}, 1\rangle]\!] \vee [\![\langle \exists \mathit{has}.\mathit{Diploma}, 1\rangle]\!]) \\ & \wedge(\neg[\![\langle \mathit{Diploma}, 1\rangle]\!] \vee [\![\langle \forall \mathit{has}^{-}.\mathit{Graduate}, 1\rangle]\!]) \\ & \wedge(\neg[\![\langle \mathit{Diploma}, 1\rangle]\!] \vee \neg[\![\langle \mathit{Graduate}, 1\rangle]\!])\end{aligned}$$

and in Fig. 3 (right) shows the OBDD representing the function $[\![\mathcal{T}]\!]_0$ obtained from $\varphi^{\mathbf{kb}}$ by conjunctively adding

$$\begin{aligned}\varphi^{\mathbf{ex}} &= \neg[\![\langle \mathit{Diploma}, 2\rangle]\!] \vee \neg[\![\mathit{has}]\!] \vee [\![\langle \exists \mathit{has}.\mathit{Diploma}, 1\rangle]\!] \text{ and} \\ \varphi^{\mathbf{uni}} &= \neg[\![\langle \forall \mathit{has}^{-}.\mathit{Graduate}, 1\rangle]\!] \vee \neg[\![\mathit{has}^{-}]\!] \vee [\![\langle \mathit{Graduate}, 2\rangle]\!].\end{aligned}$$

Then, after the first iteration of the algorithm, we arrive at an OBDD representing $[\![\mathcal{T}]\!]_1$ which is displayed in Fig. 4. This OBDD turns out to be the final result $[\![\mathcal{T}]\!]$ .

## 5 Abox Reasoning with Disjunctive Datalog

The above algorithm does not yet take any assertional information about individuals into account. Now the proof of Theorem 7 given in [2] hinges upon the fact that the constructed domino set $\mathbb{D}_{\mathcal{T}}$ induces a model of the terminology $\mathcal{T}$, and Lemma 6 states that this is indeed the *greatest* model in a certain sense. This provides some first intuition of the problems arising when Aboxes are to be added to the knowledge base: $\mathcal{ALCI}b$ knowledge bases with Aboxes do generally not have a greatest model.

We thus employ *disjunctive datalog* as a paradigm that allows us to incorporate Aboxes into the reasoning process. The basic idea is to forge a datalog program that – depending on two given individuals *a* and *b* – describes possible dominoes that may

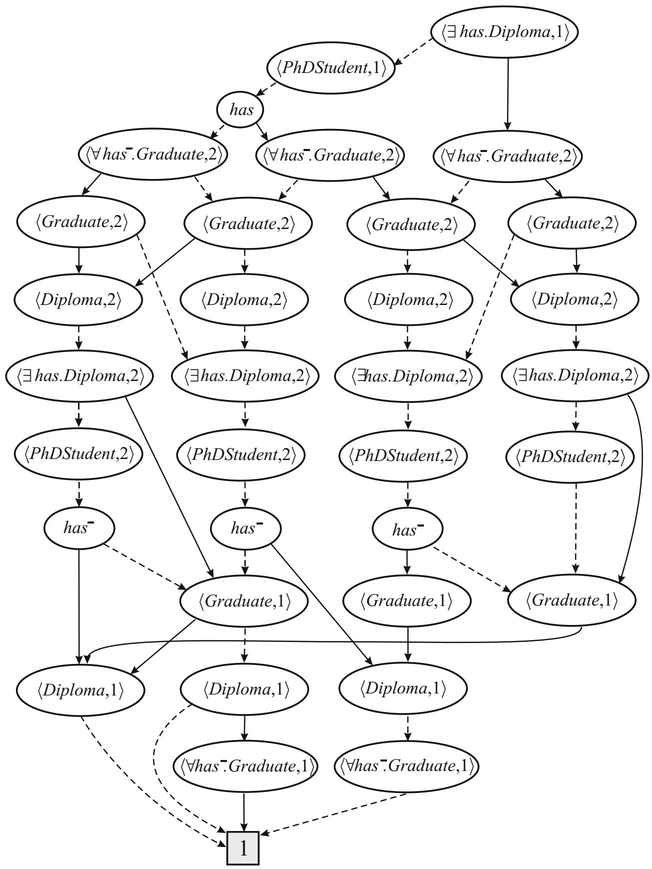


**Fig. 4.** Final OBDD obtained when processing Fig. 2, using notation as in Fig. 3. Arrows to the 0 node have been omitted for better readability.

connect *a* and *b* in models of the knowledge base. There might be various, irreconcilable such dominoes in different models, but disjunctive datalog supports such choice since it admits multiple minimal models. As long as the knowledge base has some model, there is at least one possible domino for every pair of individuals (possibly without connecting roles) – only if this is not the case, the datalog program will infer a contradiction.

In earlier sections, we have already reduced terminological reasoning in $\mathcal{ALCI}b$ to iterative constructions of Boolean formulae, and one might be tempted to directly cast these constructions into datalog. However, the terminological reasoning must take into account *all* possible individuals occurring in the constructed greatest model. If we want to represent individuals by constants in datalog, this would require us to declare exponentially many individuals in datalog. This would give up on the possible optimisation of using OBDDs, and basically just mirror the naive domino set construction in datalog.

So we use the OBDD computed from the terminology as a kind of pre-compiled version of the relevant terminological information. Abox information is then considered as a kind of incomplete specification of dominoes that must be accepted by the OBDD, and the datalog program simulates the OBDD's evaluation for each of those.

**Definition 12.** *Consider an $\mathcal{ALCIb}$ knowledge base* KB $= \langle \mathcal{A}, \mathcal{T} \rangle$ *such that $\mathcal{A}$ contains only atomic concepts, and let $\mathbb{O} = (N, n_{\mathsf{root}}, n_{\mathsf{true}}, n_{\mathsf{false}}, \mathsf{low}, \mathsf{high}, \mathsf{Var}, \lambda)$ denote an OBDD obtained as a representation of $[\![\mathsf{FLAT}(\mathcal{T})]\!]$ as in Definition 10. A disjunctive datalog program* $\mathsf{DD}(\mathrm{KB})$ *is defined as follows.* $\mathsf{DD}(\mathrm{KB})$ *uses the following predicate symbols:*

- *a unary predicate $S_C$ for every concept expression $C \in P(\mathsf{FLAT}(\mathcal{T}))$,*
- *a binary predicate $S_R$ for every atomic role $R \in \mathsf{N}_R$,*
- *a binary predicate $A_n$ for every OBDD node $n \in N$.*

*The constants in* $\mathsf{DD}(\mathrm{KB})$ *are just the individual names used in $\mathcal{A}$. The disjunctive datalog rules of* $\mathsf{DD}(\mathrm{KB})$ *are defined as follows:*

*(1)* $\mathsf{DD}(\mathrm{KB})$ *contains rules $\rightarrow A_{n_{\mathsf{root}}}(x, y)$ and $A_{n_{\mathsf{false}}}(x, y) \rightarrow$.*
*(2) If $C(a) \in \mathcal{A}$ then* $\mathsf{DD}(\mathrm{KB})$ *contains $\rightarrow S_C(a)$.*
*(3) If $R(a, b) \in \mathcal{A}$ then* $\mathsf{DD}(\mathrm{KB})$ *contains $\rightarrow S_R(a, b)$*
*(4) If $n \in N$ with $\lambda(n) = \langle C, 1\rangle$ then* $\mathsf{DD}(\mathrm{KB})$ *contains rules*
*$S_C(x) \wedge A_n(x, y) \rightarrow A_{\mathsf{high}(n)}(x, y)$ and $A_n(x, y) \rightarrow A_{\mathsf{low}(n)}(x, y) \vee S_C(x)$.*
*(5) If $n \in N$ with $\lambda(n) = \langle C, 2\rangle$ then* $\mathsf{DD}(\mathrm{KB})$ *contains rules*
*$S_C(y) \wedge A_n(x, y) \rightarrow A_{\mathsf{high}(n)}(x, y)$ and $A_n(x, y) \rightarrow A_{\mathsf{low}(n)}(x, y) \vee S_C(y)$.*
*(6) If $n \in N$ with $\lambda(n) = R$ for some $R \in \mathsf{N}_R$ then* $\mathsf{DD}(\mathrm{KB})$ *contains rules*
*$S_R(x, y) \wedge A_n(x, y) \rightarrow A_{\mathsf{high}(n)}(x, y)$ and $A_n(x, y) \rightarrow A_{\mathsf{low}(n)}(x, y) \vee S_R(x, y)$.*
*(7) If $n \in N$ with $\lambda(n) = R^-$ for some $R \in \mathsf{N}_R$ then* $\mathsf{DD}(\mathrm{KB})$ *contains rules*
*$S_R(y, x) \wedge A_n(x, y) \rightarrow A_{\mathsf{high}(n)}(x, y)$ and $A_n(x, y) \rightarrow A_{\mathsf{low}(n)}(x, y) \vee S_R(y, x)$.*

Note that the number of variables per rule in $\mathsf{DD}(\mathrm{KB})$ is bounded by 2. The semantically equivalent grounding of $\mathsf{DD}(\mathrm{KB})$ thus is a propositional program of quadratic size, and the worst-case complexity for satisfiability checking is NP, as opposed to the NExpTime complexity of disjunctive datalog in general. Note that, of course, $\mathsf{DD}(\mathrm{KB})$ may still be exponential in the size of KB in the worst case. It remains to show the correctness of the datalog translation.

**Lemma 13.** *Given an $\mathcal{ALCIb}$ knowledge base* KB *such that $\mathcal{I}$ is a model of* KB*, there is a model $\mathcal{J}$ of* $\mathsf{DD}(\mathrm{KB})$ *such that $\mathcal{I} \models C(a)$ iff $\mathcal{J} \models S_C(a)$, and $\mathcal{I} \models R(a, b)$ iff $\mathcal{J} \models S_R(a, b)$, for any $a, b \in \mathsf{N}_I$, $C \in \mathsf{N}_C$, and $R \in \mathsf{N}_R$.*

*Proof.* Let KB $= (\mathcal{A}, \mathcal{T})$. We define an interpretation $\mathcal{J}$ of $\mathsf{DD}(\mathrm{KB})$. The domain of $\mathcal{J}$ is the domain of $\mathcal{I}$, i.e. $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$. For individuals $a$, we set $a^{\mathcal{J}} := a^{\mathcal{I}}$. The interpretation of predicate symbols is now defined as follows (note that $A_n$ is defined inductively):

- $\delta \in S_C^{\mathcal{J}}$ iff $\delta \in C^{\mathcal{I}}$,
- $\langle \delta_1, \delta_2 \rangle \in S_R^{\mathcal{J}}$ iff $\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}}$,
- $\langle \delta_1, \delta_2 \rangle \in A_{n_{\mathsf{root}}}^{\mathcal{J}}$ for all $\delta_1, \delta_2 \in \Delta^{\mathcal{J}}$,
- $\langle \delta_1, \delta_2 \rangle \in A_n$ for $n \neq n_{\mathsf{root}}$ if there is a node $n'$ such that $\langle \delta_1, \delta_2 \rangle \in A_{n'}$, and one of the following is the case:

- $\lambda(n') = \langle C, i\rangle$, for some $i \in \{1, 2\}$, and $n = \mathsf{low}(n')$ and $\delta_i \notin C^{\mathcal{I}}$
- $\lambda(n') = \langle C, i\rangle$, for some $i \in \{1, 2\}$, and $n = \mathsf{high}(n')$ and $\delta_i \in C^{\mathcal{I}}$
- $\lambda(n') = R$ and $n = \mathsf{low}(n')$ and $\langle \delta_1, \delta_2\rangle \notin R^{\mathcal{I}}$
- $\lambda(n') = R$ and $n = \mathsf{high}(n')$ and $\langle \delta_1, \delta_2\rangle \in R^{\mathcal{I}}$

Mark that, in the last two items, $R$ is any role expression from $\mathsf{Var}$, and hence is a role name or its inverse. Also note that due to the acyclicity of $\mathbb{O}$, the interpretation of the $A$-predicates is indeed well-defined. We now show that $\mathcal{J}$ is a model of $\mathsf{DD}(\mathrm{KB})$. To this end, first note that the extensions of predicates $S_C$ and $S_R$ in $\mathcal{J}$ were defined to coincide with the extensions of $C$ and $R$ in $\mathcal{I}$. Since $\mathcal{I}$ satisfies $\mathcal{A}$, all ground facts of $\mathsf{DD}(\mathrm{KB})$ are satisfied by $\mathcal{J}$. This settles cases (2) and (3) of Definition 12.

Similarly, we find that the rules of cases (4)–(7) are satisfied by $\mathcal{J}$. Consider the first rule of (4), $S_C(x) \wedge A_n(x, y) \rightarrow A_{\mathsf{high}(n)}(x, y)$, and assume that $\delta_1 \in S_C^{\mathcal{J}}$ and $\langle \delta_1, \delta_2\rangle \in A_n^{\mathcal{J}}$. Thus $\delta_1 \in C^{\mathcal{I}}$, and, using the preconditions of (4), we conclude that $\langle \delta_1, \delta_2\rangle \in A_{\mathsf{high}(n)}^{\mathcal{J}}$ follows from the definition of $\mathcal{J}$. The second rule of case (4) covers the analogous negative case, and all other cases can be treated similarly.

Finally, for case (1), we need to show that $A_{n_{\mathsf{false}}}^{\mathcal{J}} = \emptyset$. For that, we first explicate the correspondence between domain elements of $\mathcal{I}$ and sets of variables of $\mathbb{O}$: Given elements $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ we define $V_{\delta_1,\delta_2} := \{\langle C, n\rangle \mid C \in P(\mathsf{FLAT}(\mathcal{T})), \delta_n \in C^{\mathcal{I}}\} \cup \{R \mid \langle \delta_1, \delta_2\rangle \in R^{\mathcal{I}}\}$, the set of variables corresponding to the $\mathcal{I}$-domino between $\delta_1$ and $\delta_2$.

Now $A_{n_{\mathsf{false}}}^{\mathcal{J}} = \emptyset$ clearly is a consequence of the following claim: for all $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ and all $n \in N$, we find that $\langle \delta_1, \delta_2\rangle \in A_n$ implies $\varphi_n(V_{\delta_1,\delta_2}) = \mathit{true}$ (using the notation of Definition 9). The proof proceeds by induction. For the case $n = n_{\mathsf{root}}$, we find that $\varphi_{n_{\mathsf{root}}} = [\![\mathcal{T}]\!]$. Since $V_{\delta_1,\delta_2}$ represents a domino of $\mathcal{I}$, the claim thus follows by combining Proposition 11 and Lemma 6.

For the induction step, let $n$ be a node such that $\langle \delta_1, \delta_2\rangle \in A_n$ follows from the inductive definition of $\mathcal{J}$ based on some predecessor node $n'$ for which the claim has already been established. Note that $n'$ may not be unique. The cases in the definition of $\mathcal{J}$ must be considered individually. Thus assume $n'$, $n$, and $\delta_1$ satisfy the first case, and that $\langle \delta_1, \delta_2\rangle \in A_n$. By induction hypothesis, $\varphi_{n'}(V_{\delta_1,\delta_2}) = \mathit{true}$, and by Definition 9 the given case yields $\varphi_n(V_{\delta_1,\delta_2}) = \mathit{true}$ as well. The other cases are similar. □

**Lemma 14.** *Given an $\mathcal{ALCI}b$ knowledge base* KB *such that $\mathcal{J}$ is a model of* $\mathsf{DD}(\mathrm{KB})$*, there is a model $\mathcal{I}$ of* $\mathsf{DD}(\mathrm{KB})$ *such that $\mathcal{I} \models C(a)$ iff $\mathcal{J} \models S_C(a)$, and $\mathcal{I} \models R(a, b)$ iff $\mathcal{J} \models S_R(a, b)$, for any $a, b \in \mathsf{N}_I$, $C \in \mathsf{N}_C$, and $R \in \mathsf{N}_R$.*

*Proof.* Let $\mathrm{KB} = (\mathcal{A}, \mathcal{T})$. We construct an interpretation $\mathcal{I}$ whose domain $\Delta^{\mathcal{I}}$ consists of all sequences starting with an individual name followed by a (possibly empty) sequence of dominoes from $\mathbb{D}_{\mathcal{T}}$ such that, for every $\delta \in \Delta^{\mathcal{I}}$,

– if $\delta$ begins with $a\langle \mathcal{A}, \mathcal{R}, \mathcal{B}\rangle$, then $\{C \mid C \in P(\mathsf{FLAT}(\mathcal{T})), a^{\mathcal{J}} \in S_C^{\mathcal{J}}\} = \mathcal{A}$, and
– if $\delta$ contains subsequent letters $\langle \mathcal{A}, \mathcal{R}, \mathcal{B}\rangle$ and $\langle \mathcal{A}', \mathcal{R}', \mathcal{B}'\rangle$, then $\mathcal{B} = \mathcal{A}'$.

For a sequence $\delta = a\langle \mathcal{A}_1, \mathcal{R}_1, \mathcal{A}_2\rangle\langle \mathcal{A}_2, \mathcal{R}_2, \mathcal{A}_3\rangle \ldots \langle \mathcal{A}_{i-1}, \mathcal{R}_{i-1}, \mathcal{A}_i\rangle$, we define $\mathrm{tail}(\delta) := \mathcal{A}_i$, whereas for a $\delta = a$ we define $\mathrm{tail}(\delta) := \{C \mid C \in P(\mathsf{FLAT}(\mathcal{T})), a^{\mathcal{J}} \in S_C^{\mathcal{J}}\}$. Now the mappings of $\mathcal{I}$ are defined as follows:

– for $a \in \mathsf{N}_I$, we have $a^{\mathcal{I}} := a$,
– for $A \in \mathsf{N}_C$, we have $\delta \in A^{\mathcal{I}}$ iff $A \in \mathrm{tail}(\delta)$,

– for $R \in \mathsf{N}_R$, we have $\langle \delta_1, \delta_2 \rangle \in R^{\mathcal{I}}$ if one of the following holds
  - $\delta_1 = a \in \mathsf{N}_I$ and $\delta_2 = b \in \mathsf{N}_I$ and $\langle a, b \rangle \in S_R^{\mathcal{J}}$, or
  - $\delta_2 = \delta_1 \langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ with $R \in \mathcal{R}$, or
  - $\delta_1 = \delta_2 \langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle$ with $\mathrm{Inv}(R) \in \mathcal{R}$.

Thus, intuitively, $\mathcal{I}$ is constructed by extracting the named individuals as well their concept (and mutual role) memberships from $\mathcal{J}$, and appending an appropriate domino-constructed tree model to each of those named individuals. We proceed by showing that $\mathcal{I}$ is indeed a model of KB.

We begin with the following auxiliary observation: For every two individual names $a, b \in \mathsf{N}_I$, and $\mathcal{R}_{ab} := \{R \mid \langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \in S_R^{\mathcal{J}}\} \cup \{\mathrm{Inv}(R) \mid \langle b^{\mathcal{J}}, a^{\mathcal{J}} \rangle \in S_R^{\mathcal{J}}\}$, the domino $\langle \mathrm{tail}(a), \mathcal{R}_{ab}, \mathrm{tail}(b) \rangle$ is contained in $\mathbb{D}_{\mathcal{T}}$ (Claim †). Using Proposition 11, it suffices to show that the Boolean function $[\![\mathcal{T}]\!]$ if applied to $V_{a,b} := \{\mathrm{tail}(a) \times \{1\} \cup \mathcal{R}_{ab} \cup \mathrm{tail}(b) \times \{2\}\}$ yields *true*. Since $[\![\mathcal{T}]\!] = \varphi_{n_{\mathrm{root}}}$, this is obtained by showing the following: For any $a, b \in \mathsf{N}_I$, we find that $\langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \in A_n^{\mathcal{J}}$ implies $\varphi_n(V_{a,b}) = \mathit{true}$. Indeed, the intended claim follows since we have $\langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \in A_{n_{\mathrm{root}}}^{\mathcal{J}}$ due to the first rule of (1) in Definition 12. We proceed by induction, starting at the leaves of the OBDD. The case $\langle a, b \rangle \in A_{n_{\mathrm{true}}}^{\mathcal{I}}$ is immediate, and $\langle a, b \rangle \in A_{n_{\mathrm{false}}}^{\mathcal{I}}$ is excluded by the second rule of (1). For the induction step, consider nodes $n, n' \in N$ such that either $\lambda(n) \in V_{a,b}$ and $n' = \mathsf{high}(n)$, or $\lambda(n) \notin V_{a,b}$ and $n' = \mathsf{low}(n)$. We assume that $\langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \in A_n^{\mathcal{J}}$, and, by induction, that the claim holds for $n'$. If $\lambda_n = \langle C, 1 \rangle$, then one of the rules of case (4) applies to $a^{\mathcal{J}}$ and $b^{\mathcal{J}}$. In both cases, we can infer $\langle a^{\mathcal{J}}, b^{\mathcal{J}} \rangle \in A_{n'}^{\mathcal{J}}$, and hence $\varphi_{n'}(V_{a,b}) = \mathit{true}$. Together with the assumptions for this case, Definition 9 implies that $\varphi_n(V_{a,b}) = \mathit{true}$ as required. The other cases are analogous.

It is easy to see that $\mathcal{I}$ satisfies all Abox axioms from KB by definition, due to the ground facts in $\mathsf{DD}(\mathrm{KB})$ (case (2) and (3) in Definition 12). To show that the Tbox is also satisfied, we need to show that all individuals of $\mathcal{I}$ are contained in the extension of each concept expression of $\mathsf{FLAT}(\mathcal{T})$. To this end, we first show that $\delta \in C^{\mathcal{I}}$ iff $C \in \mathrm{tail}(\delta)$ for all $C \in P(\mathsf{FLAT}(\mathcal{T}))$. If $C \in \mathsf{N}_C$ is atomic, this follows directly from the definition of $\mathcal{I}$. The remaining cases that may occur in $P(\mathsf{FLAT}(\mathcal{T}))$ are $C = \exists U.A$ and $C = \forall U.A$.

First consider the case $C = \exists U.A$, and assume that $\delta \in C^{\mathcal{I}}$. Thus there is $\delta' \in \Delta^{\mathcal{I}}$ with $\langle \delta, \delta' \rangle \in U^{\mathcal{I}}$ and $\delta' \in A^{\mathcal{I}}$. The construction of the domino model admits three possible cases:

– $\delta, \delta' \in \mathsf{N}_I$ and $\mathcal{R}_{\delta\delta'} \vdash U$ and $A \in \mathrm{tail}(\delta')$. Now by †, the domino $\langle \mathrm{tail}(\delta), \mathcal{R}_{\delta\delta'}, \mathrm{tail}(\delta') \rangle$ satisfies condition **ex** of Definition 5, and thus $C \in \mathrm{tail}(\delta)$ as required.
– $\delta' = \delta \langle \mathrm{tail}(\delta), \mathcal{R}, \mathrm{tail}(\delta') \rangle$ with $\mathcal{R} \vdash U$ and $A \in \mathrm{tail}(\delta')$. Since $\mathbb{D}_{\mathcal{T}} \subseteq \mathbb{D}_0$, we find that $\langle \mathrm{tail}(\delta), \mathcal{R}, \mathrm{tail}(\delta') \rangle$ satisfies condition **ex**, and thus $C \in \mathrm{tail}(\delta)$ as required.
– $\delta = \delta' \langle \mathrm{tail}(\delta'), \mathcal{R}, \mathrm{tail}(\delta) \rangle$ with $\mathrm{Inv}(\mathcal{R}) \vdash U$ and $A \in \mathrm{tail}(\delta')$. By condition **sym**, $\mathbb{D}_{\mathcal{T}}$ contains the domino $\langle \mathrm{tail}(\delta), \mathrm{Inv}(\mathcal{R}), \mathrm{tail}(\delta') \rangle$, and we can again invoke **ex** to conclude $C \in \mathrm{tail}(\delta)$.

For the converse, assume that $\exists U.A \in \mathrm{tail}(\delta)$. So $\mathbb{D}_{\mathcal{T}}$ contains a domino $\langle \mathcal{A}, \mathcal{R}, \mathrm{tail}(\delta) \rangle$. This is obvious if the sequence $\delta$ ends with a domino. If $\delta = a \in \mathsf{N}_I$, then it follows by applying † to $a$ with the first individual being arbitrary. By **sym** $\mathbb{D}_{\mathcal{T}}$ also contains the domino $\langle \mathrm{tail}(\delta), \mathcal{R}, \mathcal{A} \rangle$. By condition **delex**, the latter implies that $\mathbb{D}_{\mathcal{T}}$ contains a domino $\langle \text{tail}(\delta), \mathcal{R}', \mathcal{A}' \rangle$ such that $\mathcal{R}' \vdash U$ and $A \in \mathcal{A}'$. Thus $\delta' = \delta\langle \text{tail}(\delta), \mathcal{R}', \mathcal{A}' \rangle$ is an $\mathcal{I}$-individual such that $\langle \delta, \delta' \rangle \in U^{\mathcal{I}}$ and $\delta' \in A^{\mathcal{I}}$, and we obtain $\delta \in (\exists U.A)^{\mathcal{I}}$ as claimed.

For the second case, consider $C = \forall U.A$ and assume that $\delta \in C^{\mathcal{I}}$. As above, we find that $\mathbb{D}_{\mathcal{T}}$ contains some domino $\langle \mathcal{A}, \mathcal{R}, \text{tail}(\delta) \rangle$, where † is needed if $\delta \in \mathsf{N}_I$. By **sym** we find a domino $\langle \text{tail}(\delta), \mathcal{R}, \mathcal{A} \rangle$. For a contradiction, suppose that $\forall U.A \notin \text{tail}(\delta)$. By condition **deluni**, the latter implies that $\mathbb{D}_{\mathcal{T}}$ contains a domino $\langle \text{tail}(\delta), \mathcal{R}', \mathcal{A}' \rangle$ such that $\mathcal{R}' \vdash U$ and $A \notin \mathcal{A}'$. Thus $\delta' = \delta\langle \text{tail}(\delta), \mathcal{R}', \mathcal{A}' \rangle$ is an $\mathcal{I}$-individual such that $\langle \delta, \delta' \rangle \in U^{\mathcal{I}}$ and $\delta' \notin A^{\mathcal{I}}$. But then $\delta \notin (\forall U.A)^{\mathcal{I}}$, which is the required contradiction.

For the other direction, assume that $\forall U.A \in \text{tail}(\delta)$. According to the construction of $\mathcal{I}$, for all elements $\delta'$ with $\langle \delta, \delta' \rangle \in U^{\mathcal{I}}$, there are three possible cases:

– $\delta, \delta' \in \mathsf{N}_I$ and $\mathcal{R}_{\delta\delta'} \vdash U$. Now by †, the domino $\langle \text{tail}(\delta), \mathcal{R}_{\delta\delta'}, \text{tail}(\delta') \rangle$ satisfies condition **uni**, whence $A \in \text{tail}(\delta')$.
– $\delta' = \delta\langle \text{tail}(\delta), \mathcal{R}, \text{tail}(\delta') \rangle$ with $\mathcal{R} \vdash U$. Since $\mathbb{D}_{\mathcal{T}} \subseteq \mathbb{D}_0$, $\langle \text{tail}(\delta), \mathcal{R}, \text{tail}(\delta') \rangle$ must satisfy condition **uni**, and thus $A \in \text{tail}(\delta')$.
– $\delta = \delta'\langle \text{tail}(\delta'), \mathcal{R}, \text{tail}(\delta) \rangle$ with $\text{Inv}(\mathcal{R}) \vdash U$. By condition **sym**, $\mathbb{D}_{\mathcal{T}}$ also contains the domino $\langle \text{tail}(\delta), \text{Inv}(\mathcal{R}), \text{tail}(\delta') \rangle$, and we can again use **uni** to conclude $A \in \text{tail}(\delta')$.

Thus, $A \in \text{tail}(\delta')$ for all $U$-successors $\delta'$ of $\delta$, and hence $\delta \in (\forall U.A)^{\mathcal{I}}$ as claimed.

To finish the proof, note that any domino $\langle \mathcal{A}, \mathcal{R}, \mathcal{B} \rangle \in \mathbb{D}_{\mathcal{T}}$ satisfies condition **kb**. Using **sym**, we have that for any $\delta \in \Delta^{\mathcal{I}}$, the axiom $\bigsqcap_{D \in \text{tail}(\delta)} D \sqsubseteq C$ is a tautology for all $C \in \mathsf{FLAT}(\mathcal{T})$. As shown above, $\delta \in D^{\mathcal{I}}$ for all $D \in \text{tail}(\delta)$, and thus $\delta \in C^{\mathcal{I}}$. Hence every individual of $\mathcal{I}$ is an instance of each concept of $\mathsf{FLAT}(\mathcal{T})$ as required. □

Lemma 13 and 14 show that $\mathsf{DD}(\text{KB})$ faithfully captures both positive and negative ground conclusions of KB, and in particular that $\mathsf{DD}(\text{KB})$ and KB are equisatisfiable. As discussed in Section 2, $\mathcal{SHIQ}$ knowledge bases can be transformed into equisatisfiable $\mathcal{ALCI}b$ knowledge bases, and hence the above algorithm can also be used to decide satisfiability in the case of $\mathcal{SHIQ}$. The transformations used to convert $\mathcal{SHIQ}$ to $\mathcal{ALCI}b$, however, do not preserve all ground consequences. In particular, $\mathcal{SHIQ}$ consequences of the form $R(a, b)$ with $R$ being non-simple may not be entailed by $\mathsf{DD}(\text{KB})$. Such positive non-simple role atoms are the only case where entailments are lost, and thus $\mathsf{DD}(\text{KB})$ behaves similar to the disjunctive datalog program created by the KAON2 approach [7].

The above observation immediately allow us to add reasoning support for *DL-safe rules* [8], simply by adding the respective rules to $\mathsf{DD}(\text{KB})$ after replacing $C$ and $R$ by $S_C$ and $S_R$. A special case of this are *DL-safe* conjunctive queries, i.e. conjunctive queries that assume all variables to range only over named individuals. It is easy to see that, as a minor extension, one could generally allow for concept expressions $\forall R.A$ and $\exists R.A$ in queries and rules, simply because $\mathsf{DD}(\text{KB})$ represents these elements of $P(\mathsf{FLAT}(\mathcal{T}))$ as atomic symbols in disjunctive datalog.

## 6 Discussion

We have presented a new reasoning algorithm for $\mathcal{SHIQ}$ knowledge bases that compiles $\mathcal{SHIQ}$ terminologies into disjunctive datalog programs, which are then combined with assertional information for satisfiability checking and (ground) query answering. The approach is based on our earlier work on terminological $\mathcal{SHIQ}$ reasoning with ordered binary decision diagrams (OBDDs), which fails when introducing Aboxes as it hinges upon a form of greatest model property [2]. OBDDs now are still used to process terminologies, but are subsequently transformed into disjunctive datalog programs that can incorporate Abox data. The generation of disjunctive datalog may require exponentially many computation steps, the complexity of which depends on the concrete OBDD implementation at hand – finding *optimal encodings* is NP-complete but heuristic approximations are often used in practice. Querying the disjunctive datalog program then is co-NP-complete w.r.t. the size of the Abox, so that the data complexity of the algorithm is worst-case optimal [7].

The presented method exhibits similarities to the algorithm underlying the KAON2 reasoner [7]. In particular, pre-transformations are first applied to $\mathcal{SHIQ}$ knowledge bases, so that the resulting datalog program is not complete for querying instances of non-simple roles. Besides this restriction, extensions with DL-safe rules and ground conjunctive queries are straightforward. The presented processing, however, is very different from KAON2. Besides using OBDDs, it also employs Boolean role constructors that admit an efficient binary encoding of number restrictions [2].

For future work, the algorithm needs to be evaluated in practice. A prototype implementation was used to generate the examples within this paper, but this software is not fully functional yet. It is also evident that redundancy elimination techniques are required to reduce the number of generated datalog rules, which is also an important aspect of the KAON2 implementation. Another strand for future development is the extension of the approach to take nominals into account – significant revisions of the model-theoretic considerations are needed for that case.

## References


1. Pan, G., Sattler, U., Vardi, M.Y.: BDD-based decision procedures for the modal logic K. Journal of Applied Non-Classical Logics 16(1-2), 169–208 (2006)
2. Rudolph, S., Krötzsch, M., Hitzler, P.: Terminological reasoning in SHIQ with ordered binary decision diagrams. In: Proc. 23rd National Conference on Artificial Intelligence (AAAI 2008), pp. 529–534. AAAI Press, Menlo Park (2008)
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2007)
4. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. PhD thesis, RWTH Aachen, Germany (2001)
5. Burch, J., Clarke, E., McMillan, K., Dill, D., Hwang, L.: Symbolic model checking: $10^{20}$ states and beyond. In: Proc. 5th Annual IEEE Symposium on Logic in Computer Science, Washington, D.C., pp. 1–33. IEEE Computer Society Press, Los Alamitos (1990)
6. Huth, M.R.A., Ryan, M.D.: Logic in Computer Science: Modelling and reasoning about systems. Cambridge University Press, Cambridge (2000)
7. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. PhD thesis, Universität Karlsruhe (TH), Germany (2006)
8. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)