The Price of Selfishness: Conjunctive Query Entailment for $\mathcal{ALC}_{\mathsf{Self}}$ is $2\textsc{ExpTime}$-hard

Bartosz Bednarczyk,[1,2] **Sebastian Rudolph**[1]
[1] Computational Logic Group, Technische Universität Dresden, Germany
[2] Institute of Computer Science, University of Wrocław, Poland
{ bartosz bednarczyk, sebastian.rudolph }@tu-dresden.de

Abstract

In logic-based knowledge representation, query answering has essentially replaced mere satisfiability checking as the inferencing problem of primary interest. For knowledge bases in the basic description logic $\mathcal{ALC}$, the computational complexity of conjunctive query (CQ) answering is well known to be ExpTime-complete and hence not harder than satisfiability. This does not change when the logic is extended by certain features (such as counting or role hierarchies), whereas adding others (inverses, nominals or transitivity together with role-hierarchies) turns CQ answering exponentially harder.

We contribute to this line of results by showing the surprising fact that even extending $\mathcal{ALC}$ by just the Self operator – which proved innocuous in many other contexts – increases the complexity of CQ entailment to $2\textsc{ExpTime}$. As common for this type of problem, our proof establishes a reduction from alternating Turing machines running in exponential space, but several novel ideas and encoding tricks are required to make the approach work in that specific, restricted setting.

1 Introduction

Formal ontologies are of significant importance in artificial intelligence, playing a central role in the Semantic Web, ontology-based information integration, or peer-to-peer data management. In such scenarios, an especially prominent role is played by *description logics* (DLs) (Baader et al. 2017) – a robust family of logical formalisms used to describe ontologies and serving as the logical underpinning of contemporary standardised ontology languages. To put knowledge bases to full use as core part of intelligent information systems, much attention is being devoted to the area of ontology-based data-access, with *conjunctive queries* (CQs) being employed as a fundamental querying formalism (Ortiz and Simkus 2012).

In recent years, it has become apparent that various modelling features of DLs affect the complexity of CQ answering in a rather strong sense. Let us focus on the most popular DL, $\mathcal{ALC}$. It was first shown in (Lutz 2008) that CQ entailment is exponentially harder than the consistency problem for $\mathcal{ALC}$ extended with inverse roles ($\mathcal{I}$). Shortly after, a combination of transitivity and role-hierarchies ($\mathcal{SH}$) was shown as a culprit of higher worst-case complexity of reasoning (Eiter et al. 2009). Finally, also nominals ($\mathcal{O}$) turned out to be problematic (Ngo, Ortiz, and Simkus 2016). Nevertheless, there are also more benign DL constructs regarding the complexity of CQ entailment. Examples are counting ($\mathcal{Q}$) (Lutz 2008) (the complexity stays the same even for expressive arithmetical constraints (Baader, Bednarczyk, and Rudolph 2020)), role-hierarchies alone ($\mathcal{H}$) (Eiter, Ortiz, and Simkus 2012) or even a tamed use of higher-arity relations (Bednarczyk 2021a).

Our results. We study CQ entailment in $\mathcal{ALC}_{\mathsf{Self}}$, an extension of $\mathcal{ALC}$ with the Self operator, *i.e.* a modelling feature that allows us to specify the situation when an element is related to it*self* by a binary relationship. Among other things, this allows us to formalise the concept of a "narcissist":

$$\mathrm{Narcissist} \sqsubseteq \exists loves.\mathsf{Self}$$

or to express that no person is their own parent:

$$\mathrm{Person} \sqsubseteq \neg\exists hasParent.\mathsf{Self}.$$

The Self operator is supported by the OWL 2 Web Ontology Language and the DL $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006). Due to the simplicity of the Self operator (it only refers to one element), it is easy to accommodate for automata techniques (Calvanese, Eiter, and Ortiz 2009) or consequence-based methods (Ortiz, Rudolph, and Simkus 2010) and thus, so far, there has been no real indication that the added expressivity provided by Self may change anything, complexity-wise. Arguably, this impression is further corroborated by the observation that Self features in two profiles of OWL 2 (the EL and the RL profile), again without harming tractability (Krötzsch, Rudolph, and Hitzler 2008).

In this work, however, we show a rather counter-intuitive result, namely that CQ entailment for $\mathcal{ALC}_{\mathsf{Self}}$ is exponentially harder than for $\mathcal{ALC}$. Hence, it places the seemingly innocuous Self operator among the "malign" modelling features, like ($\mathcal{I}$), ($\mathcal{SH}$) or ($\mathcal{O}$). Moreover, this establishes $2\textsc{ExpTime}$-hardness of query entailment for the $\mathcal{Z}$ family (a.k.a. $\mathcal{ALCH}b^{\mathsf{Self}}_{\mathsf{reg}}$) of DLs (Calvanese, Eiter, and Ortiz 2009), which until now remained open as well as the $2\textsc{ExpTime}$-hardness of querying the forward guarded fragment (Bednarczyk 2021a) with equality.

Our proof goes via encoding of computation trees of alternating Turing machines working in exponential space and follows the general hardness-proof-scheme by Lutz 2008.

However, to adjust the schema to $\mathcal{ALC}_{\mathsf{Self}}$, novel ideas are required: the ability to speak about self-loops is exploited to produce a single query that traverses trees in a root-to-leaf manner and to simulate disjunction inside CQs, useful to express that certain paths are repeated inside the tree.

For space reasons, we will argue model-theoretically while refraining from presenting the axiomatisations in the main paper (they follow mostly standard ideas), but they can be found in our arXiV report (Bednarczyk and Rudolph 2021).

2 Preliminaries

We recall the basics on description logics (DLs) (Baader et al. 2017) and query answering (Ortiz and Simkus 2012).

DLs. We fix countably-infinite pairwise disjoint sets of *individual names* $\mathbf{N_I}$, *concept names* $\mathbf{N_C}$, and *role names* $\mathbf{N_R}$ and introduce the description logic $\mathcal{ALC}_{\mathsf{Self}}$. Starting from $\mathbf{N_C}$ and $\mathbf{N_R}$, the set $\mathbf{C}$ of $\mathcal{ALC}_{\mathsf{Self}}$ *concepts* is built using the following concept constructors: *negation* ($\neg \mathrm{C}$), *conjunction* ($\mathrm{C} \sqcap \mathrm{D}$), *existential restriction* ($\exists r.\mathrm{C}$), the *top concept* ($\top$), and *Self* concepts ($\exists r.\mathsf{Self}$), with the grammar:

$$\mathrm{C}, \mathrm{D} ::= \top \mid \mathrm{A} \mid \neg \mathrm{C} \mid \mathrm{C} \sqcap \mathrm{D} \mid \exists r.\mathrm{C} \mid \exists r.\mathsf{Self},$$

where $\mathrm{C}, \mathrm{D} \in \mathbf{C}$, $\mathrm{A} \in \mathbf{N_C}$, and $r \in \mathbf{N_R}$. We often employ disjunction $\mathrm{C} \sqcup \mathrm{D} := \neg(\neg\mathrm{C} \sqcap \neg\mathrm{D})$, universal restrictions $\forall r.\mathrm{C} := \neg\exists r.\neg\mathrm{C}$, bottom $\bot := \neg\top$, and the less commonly used "inline-implication" $\mathrm{C} \to \mathrm{D} := \neg\mathrm{C} \sqcup \mathrm{D}$.

Assertions are of the form $\mathrm{C}(\mathtt{a})$ or $r(\mathtt{a}, \mathtt{b})$ for $\mathtt{a}, \mathtt{b} \in \mathbf{N_I}$, $\mathrm{C} \in \mathbf{C}$, and $r \in \mathbf{N_R}$. A *general concept inclusion* (GCI) has the form $\mathrm{C} \sqsubseteq \mathrm{D}$ for concepts $\mathrm{C}, \mathrm{D} \in \mathbf{C}$. We use $\mathrm{C} \equiv \mathrm{D}$ as a shorthand for the two GCIs $\mathrm{C} \sqsubseteq \mathrm{D}$ and $\mathrm{D} \sqsubseteq \mathrm{C}$. A *knowledge base* (KB) $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ is composed of a finite non-empty set $\mathcal{A}$ (*ABox*) of assertions and a finite non-empty set $\mathcal{T}$ (*TBox*) of GCIs. We call the elements of $\mathcal{A} \cup \mathcal{T}$ *axioms*.

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
concept name	A	$\mathrm{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
conc. negation	$\neg\mathrm{C}$	$\Delta^{\mathcal{I}} \setminus \mathrm{C}^{\mathcal{I}}$
conc. intersection	$\mathrm{C} \sqcap \mathrm{D}$	$\mathrm{C}^{\mathcal{I}} \cap \mathrm{D}^{\mathcal{I}}$
exist. restriction	$\exists r.\mathrm{C}$	$\{\mathrm{d} \mid \exists \mathrm{e}.(\mathrm{d}, \mathrm{e}) \in r^{\mathcal{I}} \wedge \mathrm{e} \in \mathrm{C}^{\mathcal{I}}\}$
Self concept	$\exists r.\mathsf{Self}$	$\{\mathrm{d} \mid (\mathrm{d}, \mathrm{d}) \in r^{\mathcal{I}}\}$

Table 1: Concepts and roles in $\mathcal{ALC}_{\mathsf{Self}}$.

Axiom α	$\mathcal{I} \models \alpha$, if	
$\mathrm{C} \sqsubseteq \mathrm{D}$	$\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$	TBox $\mathcal{T}$
$\mathrm{C}(\mathtt{a})$	$\mathtt{a}^{\mathcal{I}} \in \mathrm{C}^{\mathcal{I}}$	ABox $\mathcal{A}$
$r(\mathtt{a}, \mathtt{b})$	$(\mathtt{a}^{\mathcal{I}}, \mathtt{b}^{\mathcal{I}}) \in r^{\mathcal{I}}$	

Table 2: Axioms in $\mathcal{ALC}_{\mathsf{Self}}$.

The semantics of $\mathcal{ALC}_{\mathsf{Self}}$ is defined via *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ composed of a non-empty set $\Delta^{\mathcal{I}}$ called the *domain of* $\mathcal{I}$, and an *interpretation 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 $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This mapping is extended to concepts (see Table 1) and finally used to define *satisfaction* of assertions and GCIs (see Table 2). We say that an interpretation $\mathcal{I}$ *satisfies* a KB $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ (or $\mathcal{I}$ is a *model* of $\mathcal{K}$, written: $\mathcal{I} \models \mathcal{K}$) if it satisfies all axioms of $\mathcal{A} \cup \mathcal{T}$. A KB is *consistent* (or *satisfiable*) if it has a model, and *inconsistent* (or *unsatisfiable*) otherwise.

A *homomorphism* $\mathfrak{h} : \mathcal{I} \to \mathcal{J}$ is a concept-name and role-name-preserving function that maps every element of $\Delta^{\mathcal{I}}$ to some element from $\Delta^{\mathcal{J}}$, *i.e.* we have that $\mathrm{d} \in \mathrm{A}^{\mathcal{I}}$ implies that $\mathfrak{h}(\mathrm{d}) \in \mathrm{A}^{\mathcal{J}}$ and $(\mathrm{d}, \mathrm{e}) \in r^{\mathcal{I}}$ implies $(\mathfrak{h}(\mathrm{d}), \mathfrak{h}(\mathrm{e})) \in r^{\mathcal{J}}$ for all role/concept names $r \in \mathbf{N_R}$, $\mathrm{A} \in \mathbf{N_C}$ and $\mathrm{d}, \mathrm{e} \in \Delta^{\mathcal{I}}$.

Queries. Boolean *conjunctive queries* (CQs) are conjunctions of *atoms* of the form $r(x, y)$ or $\mathrm{A}(z)$, where r is a role name, A is a concept name, and x, y, z are variables from a countably infinite set $\mathbf{N_V}$. Given a CQ q, we denote with $|q|$ the number of its atoms, and with $\mathrm{Var}(q)$ the set of all variables. Let $\mathcal{I}$ be an interpretation, q a CQ and $\pi : \mathrm{Var}(q) \to \Delta^{\mathcal{I}}$ be a variable assignment. We write $\mathcal{I} \models_{\pi} r(x, y)$ if $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$, and $\mathcal{I} \models_{\pi} \mathrm{A}(z)$ if $\pi(z) \in \mathrm{A}^{\mathcal{I}}$. We say that π is a *match* for $\mathcal{I}$ and q if $\mathcal{I} \models_{\pi} \alpha$ holds for every atom $\alpha \in q$, and that $\mathcal{I}$ *satisfies* q (denoted with: $\mathcal{I} \models q$) whenever $\mathcal{I} \models_{\pi} q$ for some match π. The definitions are lifted to KBs: q is *entailed* by a KB $\mathcal{K}$ (written: $\mathcal{K} \models q$) if every model $\mathcal{I}$ of $\mathcal{K}$ satisfies q. We stress here that satisfaction of conjunctive queries is preserved by homomorphisms, *i.e.* if $\mathcal{I} \models q$ and there is a homomorphism from $\mathfrak{h} : \mathcal{I} \to \mathcal{J}$ then $\mathcal{J} \models q$. When $\mathcal{I} \models \mathcal{K}$ but $\mathcal{I} \not\models q$, we call $\mathcal{I}$ a *countermodel* for $\mathcal{K}$ and q. The *query entailment problem* asks if $\mathcal{K} \models q$ holds for an input KB $\mathcal{K}$ and a CQ q.

Whenever convenient, we employ the *path syntax* of CQs to write queries in a concise way. By a *path expression* we mean an expression of the form

$$(\mathrm{A}_0?; r_1; \mathrm{A}_1?; r_2; \mathrm{A}_2?; \ldots; \mathrm{A}_{n-1}?; r_n; \mathrm{A}_n?)(x_0, x_n)$$

with all $r_i \in \mathbf{N_R}$, $\mathrm{A}_i \in \mathbf{N_C} \cup \{\top\}$, serving as a shorthand for

$$\bigwedge_{i=0}^{n} \mathrm{A}_i(x_i) \wedge \bigwedge_{i=1}^{n} r_i(x_{i-1}, x_i).$$

Whenever A_i happens to be $\top$, it will be removed from the expression; this does not create ambiguities. Note that path CQs are just syntactic sugar and should not be mistaken *e.g.* with regular path queries.

2.1 Alternating Turing Machines

We next fix the notation of alternating Turing machines over a binary alphabet $\{\mathtt{0}, \mathtt{1}\}$ working in exponential space (simply: ATMs). An ATM is defined as a tuple $\mathcal{M} = (\mathtt{N}, \mathtt{Q}, \mathtt{Q}_{\exists}, \mathtt{s}_I, \mathtt{s}_A, \mathtt{s}_R, \mathtt{T})$, where $\mathtt{Q}$ is a finite set of *states* (usually denoted with $\mathtt{s}$); $\mathtt{Q}_{\exists} \subseteq \mathtt{Q}$ is a set of *existential* states; $\mathtt{s}_I, \mathtt{s}_A, \mathtt{s}_R \in \mathtt{Q}$ are, respectively, pairwise different *initial*, *accepting*, and *rejecting* states; we assume that $\mathtt{s}_I \in (\mathtt{Q} \setminus \mathtt{Q}_{\exists})$. $\mathtt{T} \subseteq (\mathtt{Q} \times \{\mathtt{0}, \mathtt{1}\}) \times (\{\mathtt{0}, \mathtt{1}\} \times \mathtt{Q} \times \{-1, +1\})$ is the *transition relation*; and the natural number $\mathtt{N}$ (encoded in unary) is a parameter governing the size of the working tape. We call the states from $\mathtt{Q}_{\forall} := \mathtt{Q} \setminus \mathtt{Q}_{\exists}$ *universal*. The size of $\mathcal{M}$, denoted with $|\mathcal{M}|$, is $\mathtt{N} + |\mathtt{Q}| + |\mathtt{Q}_{\exists}| + 3 + |\mathtt{T}|$.

A *configuration* of $\mathcal{M}$ is a word $\mathtt{wsw'} \in \{\mathtt{0},\mathtt{1}\}^*\mathtt{Q}\{\mathtt{0},\mathtt{1}\}^*$ with $|\mathtt{ww'}| = 2^{\mathtt{N}}$. We call $\mathtt{wsw'}$ (i) existential (resp. universal) if $\mathtt{s}$ is existential (resp. universal), (ii) final if $\mathtt{s}$ is either $\mathtt{s}_A$ or $\mathtt{s}_R$ (iii) non-final if it is not final (iv) accepting if $\mathtt{s} = \mathtt{s}_A$. Successor configurations are defined in terms of the transition relation $\mathtt{T}$. For $\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d} \in \{\mathtt{0},\mathtt{1}\}$ and $\mathtt{v},\mathtt{v'},\mathtt{w},\mathtt{w'} \in \{\mathtt{0},\mathtt{1}\}^*$ with $|\mathtt{v}| = |\mathtt{w}|$, we let $\mathtt{wbs'w'}$ be a *quasi-successor* configuration of $\mathtt{vsav'}$ whenever $(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s'},+1) \in \mathtt{T}$, and we let $\mathtt{ws'dbw'}$ be a quasi-successor configuration of $\mathtt{vcsav'}$ whenever $(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s'},-1) \in \mathtt{T}$. If additionally we meet the requirement $\mathtt{w} = \mathtt{v}$, $\mathtt{w'} = \mathtt{v'}$, and $\mathtt{c} = \mathtt{d}$ we speak of *successor* configurations.[1]

Without loss of generality, we make the following additional assumptions about $\mathcal{M}$: First, for each non-final (*i.e.* non-accepting and non-rejecting) state $\mathtt{s}$ and every letter $\mathtt{a} \in \{\mathtt{0},\mathtt{1}\}$ the set $\mathtt{T}(\mathtt{s},\mathtt{a}) := \{(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s'},d) \in \mathtt{T}\}$ contains exactly two elements, denoted $\mathtt{T}_1(\mathtt{s},\mathtt{a})$ and $\mathtt{T}_2(\mathtt{s},\mathtt{a})$. Hence, every configuration has exactly two successor configurations. Second, for any $(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s'},d) \in \mathtt{T}$, if $\mathtt{s}$ is existential then $\mathtt{s'}$ is universal and vice versa. Third, the machine reaches a final state no later than after $2^{2^{\mathtt{N}}}$ steps (for configuration sequences). Fourth and last, $\mathcal{M}$ never attempts to move left (resp. right) on the left-most (resp. right-most) tape cell.

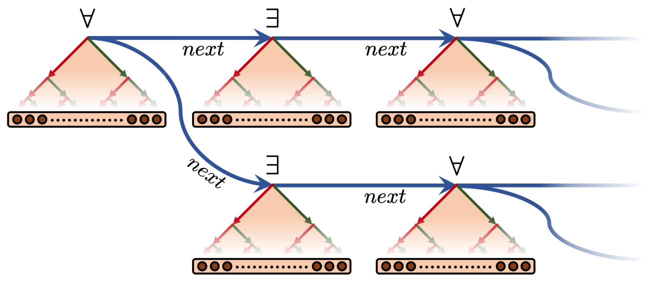
A *run* of $\mathcal{M}$ is a finite tree, with nodes labelled by configurations of $\mathcal{M}$, that satisfies all the conditions below:

- the root is labelled with the initial configuration $\mathtt{s}_I\mathtt{0}^{2^{\mathtt{N}}}$,
- each node labelled with a non-final existential configuration $\mathtt{wsw'}$ has a single child node which is labelled with one of the successor configurations of $\mathtt{wsw'}$,
- each node labelled with a non-final universal configuration $\mathtt{wsw'}$ has two child nodes which are labelled with the two successor configurations (wrt. $\mathtt{T}_1$ and $\mathtt{T}_2$) of $\mathtt{wsw'}$,
- no node labelled with a final configuration has successors.

Quasi-runs of $\mathcal{M}$ are defined analogously by replacing the notions of successors with quasi-successors. Note that every run is also a quasi-run but not vice versa.
An ATM $\mathcal{M}$ is (quasi-)*accepting* if it has an *accepting* (quasi)-run, *i.e.* one whose all leaves are labelled by accepting configurations. By (Chandra and Stockmeyer 1976) the problem of checking if a given ATM is accepting is 2EXPTIME-hard.

3 A High-Level Overview of the Encoding

Let $\mathcal{M}$ be an ATM. The core contribution of our paper is to present a polynomial-time reduction that, given $\mathcal{M}$, constructs a pair $(\mathcal{K}_{\mathcal{M}}, q_{\mathcal{M}})$ — composed of an $\mathcal{ALC}_{\mathsf{Self}}$ knowledge base and a conjunctive query — such that $\mathcal{K}_{\mathcal{M}} \not\models q_{\mathcal{M}}$ iff $\mathcal{M}$ is accepting. Intuitively, the models of $\mathcal{K}$ will encode accepting quasi-runs of $\mathcal{M}$, *i.e.* trees in which every node is a meaningful configuration of $\mathcal{M}$, but the tape contents of consecutive configurations might not be in sync as they should. The query $q_{\mathcal{M}}$ will be responsible for detecting such errors. Hence, the existence of a countermodel for $\mathcal{K}_{\mathcal{M}}$ and $q_{\mathcal{M}}$ will coincide with the existence of an accepting run of $\mathcal{M}$. The intended models of $\mathcal{K}_{\mathcal{M}}$ look as follows:



The depicted triangles are called the *configuration trees* and encode configurations of $\mathcal{M}$. The information contained in these configuration trees is "superimposed" on identical *configuration units*: full binary trees of height $\mathtt{N}{+}1$ decorated with many self-loops[2] that will provide the "navigational infrastructure" for the query $q_{\mathcal{M}}$ to detect "tape mismatches". Every such tree has $2^{\mathtt{N}}$ nodes at its $\mathtt{N}$-th level and each of these nodes represents a single tape cell of a machine. The $(\mathtt{N}{+}1)$-th level of nodes will serve a technical purpose that will be explained later. Lastly, the roots of configuration units store all remaining necessary information required for encoding: the current state of $\mathcal{M}$, the previous and the current head position as well as the transition used to arrive at this node from the previous configuration. Finally, the roots of configuration trees are interconnected by the role *next* indicating that $(\mathrm{r},\mathrm{r'}) \in \mathit{next}^{\mathcal{I}}$ holds iff the configuration represented by $\mathrm{r'}$ is a quasi-successor of the configuration of r.

4 Configuration Units

In our encoding, a vital role is played by n-*configuration units*, which will later form the backbone of configuration trees. Roughly speaking, each n-configuration unit is a full binary tree of depth n, decorated with certain concepts, roles, and self-loops. We introduce configuration units by providing the formal definition, followed by a graphical depiction and an intuitive description. In order to represent configuration units inside interpretations, we employ role names from $\mathbf{R}_{\mathit{unit}}$ as well as concept names from $\mathbf{C}_{\mathit{unit}}$:

$$\mathbf{R}_{\mathit{unit}} := \{\ell_i, r_i, \mathit{next} \mid 1 \leq i \leq n\}$$
$$\mathbf{C}_{\mathit{unit}} := \{\mathrm{Lvl}_0, \mathrm{Lvl}_i, \mathrm{L}, \mathrm{R}, \mathrm{Ad}_i^0, \mathrm{Ad}_i^1 \mid 1 \leq i \leq n\}.$$

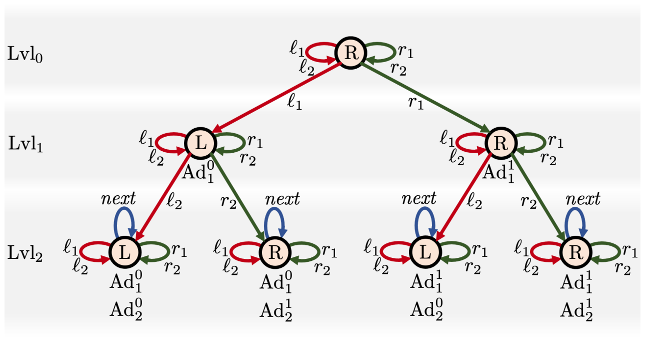
Definition 4.1 (configuration unit)**.** *Given a number n, an n-*configuration unit $\mathcal{U}$ *is an interpretation* $(\Delta^{\mathcal{U}}, \cdot^{\mathcal{U}})$ *with*

$$\begin{aligned}
\Delta^{\mathcal{U}} &= \{0,1\}^{\leq n} := \{w \in \{0,1\}^* \mid |w| \leq n\},\\
\ell_i^{\mathcal{U}} &= \{(w,w0) \mid |w| = i{-}1\} \cup \{(w,w) \mid w \in \Delta^{\mathcal{U}}\},\\
r_i^{\mathcal{U}} &= \{(w,w1) \mid |w| = i{-}1\} \cup \{(w,w) \mid w \in \Delta^{\mathcal{U}}\},\\
\mathit{next}^{\mathcal{U}} &= \{(w,w) \mid |w| = n\},\\
\mathrm{Lvl}_i^{\mathcal{U}} &= \{w \in \Delta^{\mathcal{U}} \mid |w| = i\},\\
\mathrm{L}^{\mathcal{U}} \setminus \{\varepsilon\} &= \{w0 \in \Delta^{\mathcal{U}}\} \qquad \textit{and} \qquad \mathrm{R}^{\mathcal{U}} = \Delta^{\mathcal{U}} \setminus \mathrm{L}^{\mathcal{U}},\\
(\mathrm{Ad}_i^b)^{\mathcal{U}} &= \{w \in \Delta^{\mathcal{U}} \mid |w| \geq i \textit{ and its i-th letter is } b\}.
\end{aligned}$$

[1]In words, this corresponds to the common definition of successor configurations, while for quasi-successor configurations, untouched tape cells may change arbitrarily during the transition.

[2]The concrete purpose of the abundant presence of self-loops will only become clear later, starting from Corollary 4.4.

The following drawing depicts a 2-configuration unit.



As one can see, the nodes in the tree are layered into levels according to their distance from the root. Nodes at the i-th level are members of the Lvl_i concept and their distance from the root is equal to i. Next, each non-leaf node at the i-th level has two children, the left one and the right one (satisfying, respectively, the concepts L and R) and is connected to them via the role ℓ_i and r_i, respectively. All nodes are equipped with ℓ_i- and r_i-self-loops and all leaves are additionally endowed with $next$-loops. With all nodes inside the tree, we naturally associate their addresses, *i.e.* their "numbers" when nodes from the i-th level are enumerated from left to right. In order to encode the address of a given node at the i-th level, we employ concepts $\mathrm{Ad}^b_1, \mathrm{Ad}^b_2, \ldots, \mathrm{Ad}^b_i$ with "values" b either 0 or 1, meaning that a node is in Ad^b_j iff the j-th bit of its address is equal to b. The most significant bit is Ad^b_1.

It is routine to axiomatise n-configuration units (*cf.* technical report). The provided axiomatisation is made formally precise by the following lemma:

Lemma 4.2. *There is an $\mathcal{ALC}_{\mathsf{Self}}$-KB $\mathcal{K}^n_{unit}$ such that each n-configuration unit is a model of $\mathcal{K}^n_{unit}$. For any model $\mathcal{I}$ of $\mathcal{K}^n_{unit}$ and any $\mathrm{d} \in \mathrm{Lvl}^{\mathcal{I}}_0$ there is an n-configuration unit $\mathcal{U}$ and a homomorphism $\mathfrak{h}$ from $\mathcal{U}$ into $\mathcal{I}$ with $\mathfrak{h}(\varepsilon) = \mathrm{d}$.*

At this point, we would like to give the reader some intuitions why units are decorated with different self-loops. First, we show that their presence can be exploited to navigate top-down through a given unit.

Lemma 4.3. *Let $\mathcal{U}$ be an n-configuration unit. Then for all $w \in \Delta^{\mathcal{U}}$ we have $(\varepsilon, w) \in {\ell_1}^{\mathcal{U}} \circ {r_1}^{\mathcal{U}} \circ \ldots \circ {\ell_n}^{\mathcal{U}} \circ {r_n}^{\mathcal{U}}$ with "$\circ$" denoting the composition of relations,* i.e. $s^{\mathcal{U}} \circ t^{\mathcal{U}} := \{(\mathrm{c},\mathrm{e}) \mid (\mathrm{c},\mathrm{d}) \in s^{\mathcal{U}}$ *and* $(\mathrm{d},\mathrm{e}) \in t^{\mathcal{U}}$ *for some* $\mathrm{d}\}$.

Sketch. For simplicity we use $s^{\mathcal{U}}_i$ as an abbreviation of ${\ell_1}^{\mathcal{U}} \circ {r_1}^{\mathcal{U}} \circ \ldots \circ {\ell_i}^{\mathcal{U}} \circ {r_i}^{\mathcal{U}}$. The proof is by induction, where the assumption is that for all $1 \leq i \leq n$ we have that all words w of length at most i satisfy $(\varepsilon, w) \in s^{\mathcal{U}}_i$. □

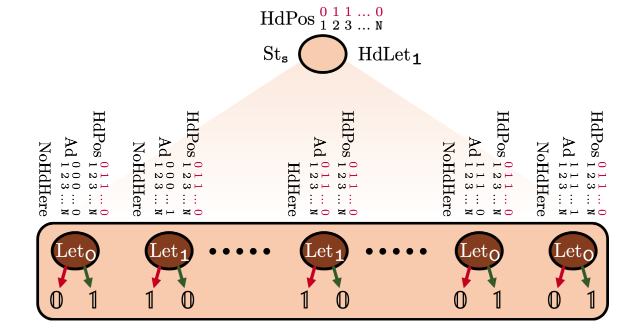
As a corollary of Lemma 4.3, we conclude that there is a *single* CQ detecting root-leaf pairs in units.

Corollary 4.4. *Let $\mathcal{U}$ be an n-configuration unit. There is a single conjunctive query q_{rl} with $x_0, x_{2n} \in \mathrm{Var}(q_{rl})$ such that the set $M = \{(\pi(x_0), \pi(x_{2n})) \mid \mathcal{U} \models_\pi q_{rl}\}$ is equal to the set of root-leaf pairs from $\mathcal{U}$,* i.e. $\mathrm{Lvl}^{\mathcal{U}}_0 \times \mathrm{Lvl}^{\mathcal{U}}_n$.

Proof. Take $q_{rl} := (\mathrm{Lvl}_0?; \ell_1; r_1; \ldots; \ell_n; r_n; \mathrm{Lvl}_n?)(x_0, x_{2n})$. The correctness follows from Lemma 4.3. □

5 From Units to Configuration Trees

In the next step, we enrich $(\mathtt{N}+1)$-configuration units with additional concepts, allowing the units to represent a meaningful configuration of our ATM $\mathcal{M} = (\mathtt{N}, \mathtt{Q}, \mathtt{Q}_\exists, \mathtt{s}_I, \mathtt{s}_A, \mathtt{s}_R, \mathtt{T})$. To this end, we employ a variety of new concept names from $\mathbf{C}_{conf}$ consisting of $\mathrm{HdHere}, \mathrm{NoHdHere}, \mathrm{St}_\mathtt{s}, \mathrm{HdPos}^b_i, \mathrm{HdLet}_\mathtt{a}, \mathrm{Let}_\mathtt{a}, \mathbb{0}, \mathbb{1}$, where $\mathtt{s} \in \mathtt{Q}, b \in \{0,1\}, i \in \{1, \ldots, \mathtt{N}\}, \mathtt{a} \in \{\mathtt{0}, \mathtt{1}\}$.



Before turning to a formal definition we first describe how configurations are structurally represented in models. Recall that a configuration of $\mathcal{M}$ is a word $\mathtt{wsw'}$ with $|\mathtt{ww'}| = 2^\mathtt{N}$ (called *tape*) and $\mathtt{s} \in \mathtt{Q}$. In our encoding, this configuration will be represented by an $(\mathtt{N}+1)$-configuration unit $\mathcal{C}$ decorated by concepts from $\mathbf{C}_{conf}$. The interpretation $\mathcal{C}$ stores the state $\mathtt{s}$, by associating the state concept $\mathrm{St}_\mathtt{s}$ to its root. The tape content $\mathtt{ww'}$ is represented by the internal nodes of $\mathcal{C}$: the i-th letter of $\mathtt{ww'}$ (*i.e.* the content of the ATM's i-th tape cell) is represented by the i-th node (according to their binary addresses) at the $\mathtt{N}$-th level. In case this letter is $\mathtt{0}$, the corresponding node will be assigned the concept $\mathrm{Let}_\mathtt{0}$, while $\mathtt{1}$ is represented by $\mathrm{Let}_\mathtt{1}$. Yet, for reasons that will become clear only later, the tape cells' content is additionally represented in another way: if it is $\mathtt{0}$, then we label the i-th node's left child with $\mathbb{0}$ and its right child with $\mathbb{1}$. The reverse situation is implemented when node represents the letter $\mathtt{1}$. Finally, there is a unique tape cell that is visited by the head of $\mathcal{M}$ and the node corresponding to this cell is explicitly marked by the concept HdHere while all other "tape cell nodes" are marked by NoHdHere. In order to implement this marking correctly, the head's position's address needs to be explicitly recorded. Consequently, $\mathcal{C}$'s root node stores this address (binarily encoded using the HdPos^b_i concepts) and from there, these concept assignments are broadcast to and stored in all tape cell nodes on the $\mathtt{N}$-th level. Similarly, we decorate $\mathcal{C}$'s root with the concept $\mathrm{HdLet}_\mathtt{a}$ meaning that the current letter scanned by the head is $\mathtt{a}$.

After this informal description and depiction, the formal definition of configuration trees should be plausible.

Definition 5.1 (configuration tree)**.** *A configuration tree $\mathcal{C}$ of $\mathcal{M}$ is an interpretation $\mathcal{C} = (\Delta^{\mathcal{C}}, \cdot^{\mathcal{C}})$ such that $\mathcal{C}$ is an $(\mathtt{N}+1)$-configuration unit additionally satisfying:*

- *There exists a unique state $\mathtt{s} \in \mathtt{Q}$ such that $(\mathrm{St}_\mathtt{s})^{\mathcal{C}} = \{\varepsilon\}$ and $(\mathrm{St}_{\mathtt{s}'})^{\mathcal{C}} = \emptyset$ for all $\mathtt{s}' \neq \mathtt{s}$.*
- $(\mathrm{Lvl}_{\mathtt{N}+1})^{\mathcal{C}} = \mathbb{0}^{\mathcal{C}} \cup \mathbb{1}^{\mathcal{C}}$ *and* $\mathbb{0}^{\mathcal{C}} \cap \mathbb{1}^{\mathcal{C}} = \emptyset$.

- $(\mathrm{Let}_{\mathtt{0}})^{\mathcal{C}} = \{w \mid w0 \in \mathbb{0}^{\mathcal{C}}, w1 \in \mathbb{1}^{\mathcal{C}}\}$, $(\mathrm{Let}_{\mathtt{1}})^{\mathcal{C}} = \{w \mid w0 \in \mathbb{1}^{\mathcal{C}}, w1 \in \mathbb{0}^{\mathcal{C}}\}$, *and* $(\mathrm{Let}_{\mathtt{0}})^{\mathcal{C}} \cup (\mathrm{Let}_{\mathtt{1}})^{\mathcal{C}} = \mathrm{Lvl}_{\mathtt{N}}^{\mathcal{C}}$.
- *There is a* unique *word* w_{head} *of length* $\mathtt{N}$ *witnessing* $\mathrm{HdHere}^{\mathcal{C}}=\{w_{head}\}$ *and* $\mathrm{NoHdHere}^{\mathcal{C}}=\mathrm{Lvl}_{\mathtt{N}}^{\mathcal{C}} \setminus \{w_{head}\}$.
- *For* $1 \leq i \leq \mathtt{N}$ *and* $b \in \{0,1\}$ *s.t.* $w_{head} \in (\mathrm{Ad}_i^b)^{\mathcal{C}}$ *we put* $(\mathrm{HdPos}_i^b)^{\mathcal{C}} = \mathrm{Lvl}_0^{\mathcal{C}} \cup \mathrm{Lvl}_{\mathtt{N}}^{\mathcal{C}}$ *and* $(\mathrm{HdPos}_i^{1-b})^{\mathcal{C}} = \emptyset$.
- $\mathrm{HdLet}_{\mathtt{a}}^{\mathcal{C}} = \{\varepsilon\}$ *and* $\mathrm{HdLet}_{\mathtt{1-a}}^{\mathcal{C}} = \emptyset$, *where* $\mathtt{a}$ *is the unique letter from* $\{\mathtt{0},\mathtt{1}\}$ *such that* $w_{head} \in \mathrm{Let}_{\mathtt{a}}^{\mathcal{C}}$.

The axiomatisation $\mathcal{K}_{conf}$ of configuration trees can be found in the technical report, together with the proof of:

Lemma 5.2. *Any configuration tree* $\mathcal{C}$ *is a model of* $\mathcal{K}_{conf}$. *For any* $\mathcal{I} \models \mathcal{K}_{conf}$ *and* $\mathrm{d} \in \mathrm{Lvl}_0^{\mathcal{I}}$ *there is a configuration tree* $\mathcal{C}$ *and a homomorphism* $\mathfrak{h}$ *from* $\mathcal{C}$ *into* $\mathcal{I}$ *with* $\mathfrak{h}(\varepsilon) = \mathrm{d}$.

6 Enriching Configuration Trees

Recall that the purpose of configuration trees is to place them inside a model that describes the run of the Turing machine $\mathcal{M}$. In particular, this will require to describe the progression of one configuration to another. In order to prepare for that, we next introduce an extended version of configuration trees that are enriched by additional information pertaining to their predecessor configuration in a run. To this end, we use new concept names from $\mathbf{C}_{enr} := \{\mathrm{PTrns}_{\mathtt{t}}, \mathrm{Init}, \mathrm{PHdHere}, \mathrm{NoPHdHere}, \mathrm{PHdAbv}, \mathrm{NoPHdAbv}, \mathrm{PHdPos}_i^b, \mathrm{PHdLet}_{\mathtt{a}}\}$, with $\mathtt{t} \in \mathtt{T}$, $1 \leq i \leq \mathtt{N}$, $b \in \{0,1\}$, and $\mathtt{a} \in \{\mathtt{0},\mathtt{1}\}$. We use $\mathbf{C}_{ptr}$ to denote the set $\{\mathrm{Init}, \mathrm{PTrns}_{\mathtt{t}} \mid \mathtt{t} \in \mathtt{T}\}$.

The concept $\mathrm{PTrns}_{\mathtt{t}}$, assigned to the root, indicates the transition, through which the configuration has been reached from the previous configuration, while Init is used as its replacement for the initial configuration. In addition, concepts PHdPos_i^b and $\mathrm{PHdLet}_{\mathtt{a}}$ are attached to the root in order to — in a way very similar to HdPos_i^b and $\mathrm{HdLet}_{\mathtt{a}}$ — indicate the previous configuration's head position as well as the letter stored in that position *on the current configuration's tape*. For the sake of our encoding we also employ the concepts PHdHere, NoPHdHere that play the role analogous to the HdHere and NoHdHere concept from configuration-trees. For technical reasons, we also introduce the concepts PHdAbv and NoPHdAbv that will label nodes on the $(\mathtt{N}{+}1)$-th level iff their parent is labelled with the corresponding concept from $\{\mathrm{PHdHere}, \mathrm{NoPHdHere}\}$.

Definition 6.1 (enriched configuration tree)**.** *An* enriched configuration tree $\mathcal{E}$ *of* $\mathcal{M}$ *is an interpretation* $\mathcal{E} = (\Delta^{\mathcal{E}}, \cdot^{\mathcal{E}})$ *such that* $\mathcal{E}$ *is a configuration tree additionally satisfying the following conditions on concepts from* $\mathbf{C}_{enr}$*:*

- *There is exactly one concept* $\mathrm{C} \in \mathbf{C}_{ptr}$ *for which* $\mathrm{C}^{\mathcal{E}} = \{\varepsilon\}$ *and for all* $\mathrm{C}' \in \mathbf{C}_{ptr} \setminus \{\mathrm{C}\}$ *we have* $(\mathrm{C}')^{\mathcal{E}} = \emptyset$.
- $\mathrm{PTrns}_{(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s}',d)}^{\mathcal{E}} = \{\varepsilon\}$ *implies* $(\mathrm{St}_{\mathtt{s}'})^{\mathcal{E}} = \{\varepsilon\}$ *for all transitions* $(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s}',d) \in \mathtt{T}$.
- $\mathrm{PHdHere}^{\mathcal{E}}=\{w_{phd}\}$ *and* $\mathrm{NoPHdHere}^{\mathcal{E}}=\mathrm{Lvl}_{\mathtt{N}}^{\mathcal{E}} \setminus \{w_{phd}\}$ *for the* $\mathtt{N}$*-digit binary word* w_{phd} *encoding*
  - *the number obtained as* $w_{head} - d$ *(see: Definition 5.1) whenever* $\mathrm{PTrns}_{(\mathtt{s},\mathtt{a},\mathtt{b},\mathtt{s}',d)}^{\mathcal{E}} = \{\varepsilon\}$*, or*
  - *the number* 0 *in case* $\mathrm{Init}^{\mathcal{E}} = \{\varepsilon\}$.
- $\mathrm{PHdAbv}^{\mathcal{E}} = \{w0, w1 \mid w \in \mathrm{PHdHere}^{\mathcal{E}}\}$ *and* $\mathrm{NoPHdAbv}^{\mathcal{E}} = \mathrm{Lvl}_{\mathtt{N}+1}^{\mathcal{E}} \setminus \mathrm{PHdAbv}^{\mathcal{E}}$.
- $(\mathrm{PHdPos}_i^b)^{\mathcal{E}} = \mathrm{Lvl}_0^{\mathcal{E}} \cup \mathrm{Lvl}_{\mathtt{N}}^{\mathcal{E}}$ *and* $(\mathrm{PHdPos}_i^{1-b})^{\mathcal{E}} = \emptyset$ *for all* $1 \leq i \leq \mathtt{N}$ *and* $0 \leq b \leq 1$ *with* $w_{phd} \in (\mathrm{Ad}_i^b)^{\mathcal{E}}$.
- $\mathrm{PHdLet}_{\mathtt{a}}^{\mathcal{E}} = \{\varepsilon\}$ *and* $\mathrm{PHdLet}_{\mathtt{1-a}}^{\mathcal{E}} = \emptyset$*, where* $\mathtt{a}$ *is the unique letter from* $\{\mathtt{0},\mathtt{1}\}$ *such that* $w_{phd} \in \mathrm{Let}_{\mathtt{a}}^{\mathcal{E}}$.
- $\mathrm{Init}^{\mathcal{E}}=\{\varepsilon\}$ *implies* $\varepsilon \in \mathrm{L}^{\mathcal{E}}$, $\mathrm{St}_{\mathtt{s}_I}^{\mathcal{E}}=\{\varepsilon\}$, $\mathrm{Let}_{\mathtt{0}}^{\mathcal{E}} = \mathrm{Lvl}_{\mathtt{N}}^{\mathcal{E}}$*, and* $\mathrm{HdPos}_i^0 = \mathrm{PHdPos}_i^0 = \mathrm{Lvl}_0^{\mathcal{E}} \cup \mathrm{Lvl}_{\mathtt{N}}^{\mathcal{E}}$ *for all* $1 \leq i \leq \mathtt{N}$.

The corresponding axiomatisation $\mathcal{K}_{enr}$ as well as the proof of the following lemma can be found in the technical report.

Lemma 6.2. *Any enriched configuration tree of* $\mathcal{E}$ *is a model of* $\mathcal{K}_{enr}$. *For any model* $\mathcal{I}$ *of* $\mathcal{K}_{enr}$ *and any* $\mathrm{d} \in \mathrm{Lvl}_0^{\mathcal{I}}$*, there is an enriched configuration tree* $\mathcal{E}$ *and a homomorphism* $\mathfrak{h}$ *from* $\mathcal{E}$ *into* $\mathcal{I}$ *with* $\mathfrak{h}(\varepsilon) = \mathrm{d}$.

7 Describing Accepting Quasi-Runs

Recall that a quasi-run $\mathfrak{R}$ of $\mathcal{M}$ is simply a tree labelled with configurations of $\mathcal{M}$ where the root is labelled with the initial configuration $\mathtt{s}_I\mathtt{0}^{2^{\mathtt{N}}}$. Each node representing an existential configuration has one child labelled with a quasi-successor configuration, while each node representing a universal configuration has two children labelled by quasi-successor configurations obtained via different transitions.

In order to represent an accepting quasi-run by a model, we employ the notion of a *quasi-computation tree* $\mathcal{Q}$, a structure intuitively defined from some $\mathfrak{R}$ as follows: replace every node of $\mathfrak{R}$ by its corresponding configuration tree, adequately enriched with information about its generating transition and the predecessor configuration. The roots of these enriched configuration trees are linked via the $next$ role to express the quasi-succession relation of $\mathfrak{R}$. The roots of enriched configuration trees representing universal configurations are chosen to be labelled with L, their left $next$-child with L and their right $next$-child with R (both corresponding to existential configurations). As expected, the Init concept decorates the root of the distinguished enriched configuration tree that represents $\mathfrak{R}$'s initial configuration. As our attention is restricted to *accepting* quasi-runs $\mathfrak{R}$, we require that no enriched configuration tree occurring in $\mathcal{Q}$ carries a rejecting state. We now give a formal definition of such a structure $\mathcal{Q}$.

Definition 7.1 (quasi-computation tree)**.** *A* quasi-computation tree $\mathcal{Q}$ *of* $\mathcal{M}$ *is an interpretation* $\mathcal{Q} = (\Delta^{\mathcal{Q}}, \cdot^{\mathcal{Q}})$ *satisfying the following properties:*

- $\Delta^{\mathcal{Q}} := \mathfrak{T} \times \{0,1\}^{\leq \mathtt{N}+1}$*, where* $\mathfrak{T}$ *is*[3] *a prefix-closed subset of* $\{\mathtt{10},\mathtt{00}\}^* \cdot \{\varepsilon,\mathtt{0},\mathtt{1}\}$ *with* $\mathfrak{w}\mathtt{1} \in \mathfrak{T}$ *implying* $\mathfrak{w}\mathtt{0} \in \mathfrak{T}$.
- *For every* $\mathfrak{w} \in \mathfrak{T}$*, the substructure of* $\mathcal{Q}$ *induced by* $\{\mathfrak{w}\} \times \{0,1\}^{\leq \mathtt{N}+1}$ *is isomorphic to an enriched configuration tree of* $\mathcal{M}$ *via the isomorphism* $(\mathfrak{w}, w) \mapsto w$.
- $(\varepsilon, \mathfrak{w}) \in \mathrm{R}^{\mathcal{Q}}$ *if* $\mathfrak{w}$ *ends with* $\mathtt{1}$*, otherwise* $(\varepsilon, \mathfrak{w}) \in \mathrm{L}^{\mathcal{Q}}$.
- *For any* $\mathfrak{w} \neq \mathfrak{w}'$ *and arbitrary* $w, w' \in \{0,1\}^{\leq \mathtt{N}+1}$ *holds* $((\mathfrak{w}, w), (\mathfrak{w}', w')) \notin s^{\mathcal{Q}}$ *for any* $s \in \mathbf{R}_{unit} \setminus \{next\}$.

[3]$\mathfrak{T}$ is just a binary tree in which nodes at the i-th level have exactly 2 children if i is even and exactly one child otherwise.

- $next^{\mathcal{Q}} \setminus \{(\mathrm{d},\mathrm{d}) \mid \Delta^{\mathcal{Q}}\times\Delta^{\mathcal{Q}}\} = \{((\mathfrak{w},\varepsilon),(\mathfrak{w}\mathfrak{b},\varepsilon)) \mid \mathfrak{w}\mathfrak{b},\mathfrak{w}\in\mathfrak{T},\mathfrak{b}\in\{\mathfrak{o},\mathfrak{1}\}\}$.
- $\mathrm{Init}^{\mathcal{Q}} = \{(\varepsilon,\varepsilon)\}$.
- *For any* $\mathfrak{w}\mathfrak{o}\in\mathfrak{T}$ *with* $(\mathfrak{w},\varepsilon)\in\mathrm{St}^{\mathcal{Q}}_{\mathtt{s}}$ *and* $(\mathfrak{w},\varepsilon)\in\mathrm{Let}^{\mathcal{Q}}_{\mathtt{a}}$
  - *if* $\mathfrak{w}\mathfrak{1}\in\mathfrak{T}$ *then* $(\mathfrak{w}\mathfrak{o},\varepsilon)\in\mathrm{PTrns}^{\mathcal{Q}}_{\mathtt{T}_1(\mathtt{s},\mathtt{a})}$ *and* $(\mathfrak{w}\mathfrak{1},\varepsilon)\in\mathrm{PTrns}^{\mathcal{Q}}_{\mathtt{T}_2(\mathtt{s},\mathtt{a})}$*,*
  - *if* $\mathfrak{w}\mathfrak{1}\notin\mathfrak{T}$ *then* $(\mathfrak{w}\mathfrak{o},\varepsilon)\in\mathrm{PTrns}^{\mathcal{Q}}_{\mathtt{T}_1(\mathtt{s},\mathtt{a})}$ *or* $(\mathfrak{w}\mathfrak{o},\varepsilon)\in\mathrm{PTrns}^{\mathcal{Q}}_{\mathtt{T}_2(\mathtt{s},\mathtt{a})}$*.*
- *If* $(\mathfrak{w},w)\in\mathrm{HdHere}^{\mathcal{Q}}$ *and* $\mathfrak{w}\mathfrak{b}\in\mathfrak{T}$ *then* $(\mathfrak{w}\mathfrak{b},w)\in\mathrm{PHdHere}^{\mathcal{Q}}$*.*
- $\mathrm{St}^{\mathcal{Q}}_{\mathtt{s}_R}=\emptyset$ *as well as* $(\mathfrak{w},\varepsilon)\in\mathrm{St}^{\mathcal{Q}}_{\mathtt{s}_A}$ *iff* $\mathfrak{w}\in\mathfrak{T}$ *and* $\mathfrak{w}\mathfrak{o}\notin\mathfrak{T}$*.*

Let $\mathcal{T}_{\mathcal{M}}$ be the set of all GCIs presented so far (plus the additional ones used to axiomatise quasi-computations) and let $\mathcal{A}_{\mathcal{M}}$ be an ABox composed of a single axiom $\mathrm{Init}(\mathtt{a})$ for a fresh individual name $\mathtt{a}$. Put $\mathcal{K}_{\mathcal{M}} := (\mathcal{A}_{\mathcal{M}},\mathcal{T}_{\mathcal{M}})$.

Lemma 7.2. *Any accepting quasi-computation tree $\mathcal{Q}$ of $\mathcal{M}$ is a model of $\mathcal{K}_{\mathcal{M}}$. For any model $\mathcal{I}$ of $\mathcal{K}_{\mathcal{M}}$ there exists an accepting quasi-computation tree $\mathcal{Q}$ and a homomorphism $\mathfrak{h}:\mathcal{Q}\to\mathcal{I}$ with $\mathfrak{h}(\varepsilon,\varepsilon)=a^{\mathcal{I}}$.*

8 Detecting Faulty Runs with a Single CQ

We finally have reached the point where querying comes into play. Our last goal is to design *one single* conjunctive query that detects "faulty configuration progressions" in quasi-computation trees, meaning that it matches a pair of two positions in consecutive configuration trees representing the same cell and being untouched by the head of $\mathcal{M}$ yet storing different letters. Note that the lack of such cells in a quasi-computation tree means that any two consecutive configuration trees represent not only quasi-successor configuration but actually proper successors and hence the structure as such even represents a "proper" run. We start by formalising our requirements to such a query:

Lemma 8.1. *There exists a CQ $q_{\mathcal{M}}$ of size polynomial in $\mathtt{N}$ with two distinguished variables x,y such that for all quasi-computation trees $\mathcal{Q}$ we have $\mathcal{Q}\models_\pi q_{\mathcal{M}}$ iff there exists a word $\mathfrak{w}$, a letter $\mathfrak{b}$ and a word w of length $\mathtt{N}{+}1$ such that:*

- $\pi(x)=(\mathfrak{w},w)$*,* $\pi(y)=(\mathfrak{w}\mathfrak{b},w)$*,*
- $\pi(y)\in\mathrm{NoPHdAbv}^{\mathcal{Q}}$*,*
- $\pi(x)\in\mathbb{0}^{\mathcal{Q}}$ *and* $\pi(y)\in\mathbb{1}^{\mathcal{Q}}$*.*

Note the asymmetry in the 3rd bullet point above – we ignore the reverse constellation. Yet, due to our encoding if the reverse situation occurs then so does the original one. Hence, every mismatch in a sense causes two inconsistencies from the point of $\mathtt{N}{+}1$-level nodes. This solves the mystery of introducing level $\mathtt{N}{+}1$ in our configuration trees and the particular encoding of tape symbols: it is crucial for catching faulty progressions by using one single CQ. Before proving Lemma 8.1 we show how it implies our main theorem:

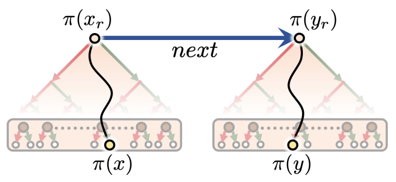
Theorem 8.2. *Conjunctive query entailment over $\mathcal{ALC}_{\mathsf{Self}}$ knowledge bases is* 2EXPTIME*-hard.*

Proof. It suffices to show that CQ non-entailment over $\mathcal{ALC}_{\mathsf{Self}}$ KBs is 2EXPTIME-hard. Take $\mathcal{K}_{\mathcal{M}}$ as defined in Section 7 and $q_{\mathcal{M}}$ as given by Lemma 8.1. We claim that $\mathcal{K}_{\mathcal{M}}\not\models q_{\mathcal{M}}$ iff $\mathcal{M}$ is accepting. The "if" direction is easy: we take an accepting run of $\mathcal{M}$ and turn it into quasi-computation tree $\mathcal{Q}$. By Lemma 7.2 we conclude that $\mathcal{Q}\models\mathcal{K}_{\mathcal{M}}$. We also have that $\mathcal{Q}\not\models q$ due to the fact that any two consecutive configuration trees represent proper successor configurations. For the second direction it suffices to show that if $\mathcal{M}$ is not accepting then $\mathcal{K}_{\mathcal{M}}\models q_{\mathcal{M}}$. Indeed, assume that $\mathcal{M}$ is not accepting and let $\mathcal{I}$ be a model of $\mathcal{K}_{\mathcal{M}}$. By Lemma 7.2 there is a quasi-computation tree $\mathcal{Q}$ and a homomorphism $\mathfrak{h}:\mathcal{Q}\to\mathcal{I}$ with $\mathfrak{h}(\varepsilon,\varepsilon)=\mathrm{a}^{\mathcal{I}}$. But this quasi-computation tree must represent a "faulty" run – in the opposite case it would correspond to an accepting run of $\mathcal{M}$, which does not exist by assumption. Hence there must be a match of $q_{\mathcal{M}}$ to $\mathcal{Q}$. As query matches are preserved under homomorphisms, we conclude $\mathcal{I}\models q_{\mathcal{M}}$. Thus all models $\mathcal{I}$ of $\mathcal{K}_{\mathcal{M}}$ have matches of $q_{\mathcal{M}}$, which implies $\mathcal{K}_{\mathcal{M}}\models q_{\mathcal{M}}$. □

In the forthcoming query definitions, we employ a convenient naming scheme. By writing $q[x,y]$ we indicate that the variables $x,y\in\mathrm{Var}(q)$ are *global* (*i.e.* the same across (sub)queries that we might join together) while its other variables are *local* (*i.e.* pairwise different from any variables occurring in other queries — this can always be enforced by renaming). Going back to the query, we proceed as follows. We first prepare a query $q_{main}[x,y]$ with two global distinguished variables x,y that relates any two domain elements whenever they are leaf nodes of consecutive computation trees. Then $q_{main}[x,y]$ is combined with queries $q^{i}_{adr}[x,y]$ for all $1\leq i\leq\mathtt{N}{+}1$ with the intended meaning that x and y have the same i-th bit of their addresses. Additionally, our final query will require that x be mapped to a node satisfying $\mathbb{0}$ and y to a node satisfying $\mathbb{1}$ and $\mathrm{NoPHdHere}$.

To construct $q_{main}[x,y]$ we essentially employ Lemma 4.3.

Lemma 8.3. *There exists a CQ $q_{main}[x,y]$ such that for any quasi-computation tree $\mathcal{Q}$ the set $M_{q_{main}}:=\{(\pi(x),\pi(y))\mid\mathcal{Q}\models_\pi q_{main}\}$ is composed precisely of any pair of leaves of two consecutive configuration trees of $\mathcal{Q}$. Formally: $M_{q_{main}}=\{((\mathfrak{w},w),(\mathfrak{w}\mathfrak{b},v))\in\Delta^{\mathcal{Q}}\mid |w|=|v|=\mathtt{N}{+}1,\mathfrak{b}\in\{0,1\}\}$.*



Proof. It suffices to take $q_{main}:=q_{rl}[x_r,x]\wedge next(x_r,y_r)\wedge q_{rl}[y_r,y]$. Let $\mathcal{Q}\models_\pi q_{main}$. That $M_{q_{main}}$ is a superset of the set above follows from the fact that quasi-computation trees are computation units and hence, containment follows by Corollary 4.4. We now focus on the other direction. Note that by the 5th item of Definition 7.1 we know that $\pi(x_r)$ and $\pi(y_r)$ must be two distinct roots of enriched configuration trees $\mathcal{E}_{x_r},\mathcal{E}_{y_r}$. By the 4th item of Definition 7.1 we know that the interpretation of the rs and ℓs is restricted to pairs of domain elements located inside the same enriched configuration tree (and by their definition to configuration trees and by their definition to configuration units). Since q_{rl} only employs the roles ℓ_i,r_i and the concepts $\mathrm{Lvl}_0,\mathrm{Lvl}_{\mathtt{N}+1}$ we conclude that

q_{rl} has exactly the same set of matches in $\mathcal{E}_{x_r}$ as in its underlying unit. Hence, by Corollary 4.4 we know that x (resp. y) is indeed mapped to a leaf of $\mathcal{E}_{x_r}$ (resp. to a leaf of $\mathcal{E}_{y_r}$), which finishes the proof. $\square$

The next part of our query construction focuses on subqueries $q^i_{adr}[x, y]$ that are meant to relate leaves having equal i-th bits of addresses. In order to construct it we combine together several smaller queries, written in path syntax below.

- We let $q_{\downarrow}[x, y] := (\ell_1; r_1; \ldots; \ell_{\mathtt{N}+1}; r_{\mathtt{N}+1})(x, y)$ define the *top-down query*. It intuitively traverses an enriched configuration tree in a top-down manner. Note that $q_{\downarrow}[x, y]$ is actually the major sub-query of $q_{rl}[x, y]$.
- The *ℓ_i-top-down query* $q_{\ell i\downarrow}[x, y]$ is similar to $q_{\downarrow}[x, y]$, but with the $\ell_i; r_i$ part replaced by just ℓ_i. The intended behavior is that again a tree is traversed from root to leaves, but this time, an ℓ_i edge must be crossed when going from the $(i-1)$-th to the i-th level. The *r_i-top-down query* $q_{ri\downarrow}[x, y]$ is defined by replacing $\ell_i; r_i$ in $q_{\downarrow}[x, y]$ with r_i.

An important ingredient in the construction is the query $q^{i\text{-th bit}}_{=0}[x, y]$ defined as follows:

$$\mathrm{Lvl}_{\mathtt{N}+1}(x) \wedge q_{\ell i\downarrow}[x',x] \wedge next(x',y') \wedge q_{\ell i\downarrow}[y',y] \wedge \mathrm{Lvl}_{\mathtt{N}+1}(y).$$

In total analogy, we define $q^{i\text{-th bit}}_{=1}[x, y]$ by using $q_{ri\downarrow}$ instead of $q_{\ell i\downarrow}$. Any match π of the query $q^{i\text{-th bit}}_{=b}[x, y]$ instantiates the variables x and y in a quasi-computation tree $\mathcal{Q}$ according to one of the following two scenarios: either $\pi(x) = \pi(y)$ or $\pi(x)$ and $\pi(y)$ are leaves in two consecutive enriched configuration trees inside the quasi-computation tree and both of these leaves have their i-th address bit set to b.

Lemma 8.4. *Let $\mathcal{Q}$ be a quasi-computation tree and let $M_{q^{i\text{-th bit}}_{=b}} = \{(\pi(x),\pi(y)) \mid \mathcal{Q} \models_{\pi} q^{i\text{-th bit}}_{=b}\}$ for $b \in \{0,1\}$. Then $M_{q^{i\text{-th bit}}_{=b}}$ is equal to the union of $M^b_1 := \{((\mathfrak{w}, w), (\mathfrak{w}, w))\}$ and $M^b_2 := \{((\mathfrak{w}, ubv), (\mathfrak{w}\mathfrak{b}, u'bv')) \mid |u|=|u'|=i-1\}$.*

Proof. We show the statement for $b = 0$, the case for $b = 1$ then follows by symmetry. First we show $M^0_1 \subseteq M_{q^{i\text{-th bit}}_{=0}}$. This is easy: for any leaf $\mathrm{d} = (\mathfrak{w}, w)$ we map all variables of $q^{i\text{-th bit}}_{=0}[x, y]$ into d; this is a match due to the presence of all the self-loops at the leaves. To show $M^0_2 \subseteq M_{q^{i\text{-th bit}}_{=0}}$ we take any $\mathrm{d} = (\mathfrak{w}, w)$ and $\mathrm{e} = (\mathfrak{w}\mathfrak{b}, v)$. Let π be a variable assignment that maps x to d, y to e, x' to $(\mathfrak{w}, \varepsilon)$, y' to $(\mathfrak{w}\mathfrak{b}, \varepsilon)$. The variables of $q_{\ell i\downarrow}[x', x]$ are mapped to $(\mathfrak{w}, w_j)$, where w_j is the prefix of w of length j following the path from $(\mathfrak{w}, \varepsilon)$ to $(\mathfrak{w}, w)$ level-by-level. We stress that $((\mathfrak{w}, w_{i-1}), (\mathfrak{w}, w_i)) \in \ell^{\mathcal{Q}}_i$ holds, which is crucial for the construction to work and that every $(\mathfrak{w}, w_j)$ node has all ℓ- and r-loops. The variables of $q_{\ell i\downarrow}[y', y]$ are mapped analogously. After noticing that $\mathrm{d}, \mathrm{e} \in \mathrm{Lvl}^{\mathcal{Q}}_{\mathtt{N}+1}$ and that $(\pi(x'), \pi(y')) \in next^{\mathcal{Q}}$ holds, we conclude that π is clearly a match of $q^{i\text{-th bit}}_{=0}[x, y]$ to $\mathcal{Q}$.

Now we focus on showing that $M_{q^{i\text{-th bit}}_{=0}[x,y]} \subseteq M^0_1 \cup M^0_2$. Take any match π and note that x, y must be mapped to leaves. For $\pi(x')$ and $\pi(y')$ we consider the two cases:

1. $\pi(x') = \pi(y')$. As the roots do not have $next$-loops, $\pi(x')$ must be a leaf. This implies that all variables of $q_{\ell i\downarrow}[x', x]$ map into a single domain element (otherwise we would not reach a leaf after traversing such path). Arguing similarly we infer that all variables of $q_{\ell i\downarrow}[y', y]$ are mapped to the same element. Thus $\pi(x) = \pi(y)$ holds.
2. $\pi(x') \neq \pi(y')$. Since all incoming $next$ roles from leaves are self-loops, we conclude that $\pi(x')$ is the root of some enriched quasi-computation tree and $\pi(y')$ is the root of some corresponding quasi-successor in $\mathcal{Q}$ (by the definition of $next^{\mathcal{Q}}$). By the satisfaction of $q_{\ell i\downarrow}[x', x]$ there is a sequence of domain elements contributing to a path from $\pi(x')$ to $\pi(x)$ witnessing its satisfaction. Moreover, note that since the subquery $q_{\ell i\downarrow}[x', x]$ leads from the root to a leaf it implies that we necessarily cross the ℓ_i role at the $i-1$-th level, meaning that the i-th bit of the address of $\pi(x)$ is equal to 0. Thus we infer that $\pi(x) \in (\mathrm{Ad}^0_i)^{\mathcal{Q}}$. Analogously, we infer $\pi(y) \in (\mathrm{Ad}^0_i)^{\mathcal{Q}}$. $\square$

The query $q^i_{adr}[x, y]$ pairing leaves in consecutive enriched conf. trees with coinciding i-th address bit is defined as:

$$q^i_{adr}[x, y] := q_{main}[x, y] \wedge q^{i\text{-th bit}}_{=0}[x, z] \wedge q^{i\text{-th bit}}_{=1}[z, y]$$

Lemma 8.5. *For any quasi-computation tree $\mathcal{Q}$ we have that $M_{q^i_{adr}} = \{(\pi(x), \pi(y)) \mid \mathcal{Q} \models_{\pi} q^i_{adr}[x, y]\}$ is composed precisely of the leaf pairs in two consecutive enriched configuration trees of $\mathcal{Q}$ having equal i-th bit of address, formally: $M_{q^i_{adr}} = M_{q_{main}} \cap \Big(\big({\mathrm{Ad}^0_i}^{\mathcal{Q}} \times {\mathrm{Ad}^0_i}^{\mathcal{Q}}\big) \cup \big({\mathrm{Ad}^1_i}^{\mathcal{Q}} \times {\mathrm{Ad}^1_i}^{\mathcal{Q}}\big)\Big)$.*

Sketch. By employing the definition of the query, Lemma 8.4 and relational calculus. $\square$

We are finally ready to present our query by means of which we can conclude with the proof of Lemma 8.1.

$$q_{\mathcal{M}} := \bigwedge_{i=1}^{\mathtt{N}+1} q^i_{adr}[x, y] \wedge \mathrm{NoPHdAbv}(y) \wedge \mathbb{0}(x) \wedge \mathbb{1}(y)$$

Proof of Lemma 8.1. Let $q_{\mathcal{M}}$ as defined above and observe that its size is clearly polynomial in $\mathtt{N}$. Note that $q_{\mathcal{M}}$ satisfies our requirements: The 1st item follows from two lemmas: the fact that x and y are mapped to leaves of two consecutive enriched configuration trees follows from Lemma 8.3 and the fact that x and y are mapped to nodes having equal addresses follows from Lemma 8.5 applied for every $1 \leq i \leq \mathtt{N}+1$. The 2nd and the 3rd points hold since we supplemented our query with $\mathrm{NoPHdAbv}(y) \wedge \mathbb{0}(x) \wedge \mathbb{1}(y)$. $\square$

9 Conclusions

Conjunctive query entailment for $\mathcal{ALC}_{\mathsf{Self}}$ is, in fact 2EXPTIME-complete, where membership follows from much stronger logics (Calvanese, Eiter, and Ortiz 2009). Hardness, shown in this paper, came as a quite surprise to us (in fact, we spent quite some time trying to prove EXPTIME-membership, see: (Bednarczyk 2021b)). The key insight of our proof (and maybe the take-home message from this paper) is that the presence of Self allows us to mimic case distinction over paths (and hence the handling of disjunctive information) through concatenation, by providing the opportunity for one of the two disjuncts to idle by "circling in place".

On a last note, our result holds for plain $\mathcal{ALC}_{\mathsf{Self}}$ TBoxes, since the only ABox assertion $\mathrm{Init}(\mathrm{a})$ can be replaced by the GCI $\top \sqsubseteq \exists aux.\mathrm{Init}$ for an auxiliary role name aux.

Acknowledgements

This work was supported by the ERC through the Consolidator Grant No. 771779.

References

Baader, F.; Bednarczyk, B.; and Rudolph, S. 2020. Satisfiability and Query Answering in Description Logics with Global and Local Cardinality Constraints. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020*, volume 325, 616–623. IOS Press.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press. ISBN 978-0-521-69542-8.

Bednarczyk, B. 2021a. Exploiting Forwardness: Satisfiability and Query-Entailment in Forward Guarded Fragment. In Faber, W.; Friedrich, G.; Gebser, M.; and Morak, M., eds., *Logics in Artificial Intelligence - 17th European Conference, JELIA 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12678 of *Lecture Notes in Computer Science*, 179–193. Springer.

Bednarczyk, B. 2021b. Lutz's Spoiler Technique Revisited: A Unified Approach to Worst-Case Optimal Entailment of Unions of Conjunctive Queries in Locally-Forward Description Logics. *CoRR*, abs/2108.05680.

Bednarczyk, B.; and Rudolph, S. 2021. The Price of Selfishness: Conjunctive Query Entailment for ALCSelf is 2ExpTime-hard. *CoRR*, abs/2106.15150.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2009. Regular Path Queries in Expressive Description Logics with Nominals. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 714–720.

Chandra, A. K.; and Stockmeyer, L. J. 1976. Alternation. In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976*, 98–108. IEEE Computer Society.

Eiter, T.; Lutz, C.; Ortiz, M.; and Simkus, M. 2009. Query Answering in Description Logics with Transitive Roles. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 759–764.

Eiter, T.; Ortiz, M.; and Simkus, M. 2012. Conjunctive query answering in the description logic SH using knots. *J. Comput. Syst. Sci.*, 78(1): 47–85.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The Even More Irresistible SROIQ. In Doherty, P.; Mylopoulos, J.; and Welty, C. A., eds., *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, 57–67. AAAI Press.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2008. ELP: Tractable Rules for OWL 2. In Sheth, A. P.; Staab, S.; Dean, M.; Paolucci, M.; Maynard, D.; Finin, T. W.; and Thirunarayan, K., eds., *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, 649–664. Springer.

Lutz, C. 2008. The Complexity of Conjunctive Query Answering in Expressive Description Logics. In Armando, A.; Baumgartner, P.; and Dowek, G., eds., *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, 179–193. Springer.

Ngo, N.; Ortiz, M.; and Simkus, M. 2016. Closed Predicates in Description Logics: Results on Combined Complexity. In Baral, C.; Delgrande, J. P.; and Wolter, F., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, 237–246. AAAI Press.

Ortiz, M.; Rudolph, S.; and Simkus, M. 2010. Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press.

Ortiz, M.; and Simkus, M. 2012. Reasoning and Query Answering in Description Logics. In Eiter, T.; and Krennwallner, T., eds., *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings*, volume 7487 of *Lecture Notes in Computer Science*, 1–53. Springer.