An Upper Bound for Provenance in $\mathcal{ELH}^r$ [⋆]

Rafael Peñaloza

University of Milano-Bicocca, Italy
rafael.penaloza@unimib.it

Abstract. We investigate the entailment problem in $\mathcal{ELH}^r$ ontologies annotated with provenance information. In more detail, we show that subsumption entailment is in NP if provenance is represented with polynomials from the Trio semiring and in PTime if the semiring is not commutative. The proof is based on the construction of a weighted tree automaton which recognises a language that matches with the corresponding provenance polynomial.

1 Introduction

The study of provenance has recently gained interest in description logics as a manner to keep track of the sources that are responsible for a consequence to follow from an ontology [4, 6]. The basic idea behind provenance is to assign a unique label to each axiom in an ontology, and obtain a *summary* of the causes for deriving a consequence through two operators from a semiring: a *product*, which combines together the axioms used in one derivation, and a *sum* which accumulates the products from the different possible derivations. These two operations must satisfy some properties, forming a *semiring*.

Although the motivation and the basic underlying structure is reminiscent of axiom pinpointing [3, 19], there are subtle but important differences which warrant further analysis. A primary difference is that provenance does not require minimality of the information provided (as opposed to the notion of *justification*), but still requires a coherence between the provenance elements forming a so-called provenance *monomial*; the *product* of variables identifying the axioms needed to derive a desired consequence. In addition, work in provenance is usually pursued in an abstract form, studying the properties based on a general semiring, which can later be instantiated to specific algebraic structures depending on the application. Axiom pinpointing can indeed be obtained by instantiating to a very specific semiring.

Very recently, the problem of answering provenance queries in the description logic $\mathcal{ELH}^r$ was studied [5]. That work focused on a semiring where the product operation is commutative and idempotent, and expressed the provenance information through an expanded polynomial; that is, a sum of monomials. One of the main results was a consequence-based algorithm for the monomial of an entailment problem; that is, deciding whether the provenance polynomial for a consequence contains a given monomial m. It was shown that this problem is in PSpace, but the best matching lower bound was the polynomial hardness for reasoning in $\mathcal{EL}$.

In this paper we improve that upper bound by showing that the monomial for a subsumption problem is in NP. To achieve this goal, we view the completion algorithm from [5] as a weighted tree automaton, which accepts all the completion-like proofs of a derivation. Through the behaviour of this automaton, the monomial problem is reduced to a membership problem in regular languages. To preserve polynomiality in the behaviour computation, we adapt the notion of structure sharing to automata construction with the help of *acyclic recursive automata* (also known as *hierarchical state machines* [20]). These automata are exponentially more succinct than NFA, but not more expressive, and retain most of the complexity properties of NFA.

2 Preliminaries

A *semiring* is an algebraic structure $\mathbb{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where $\oplus$ and $\otimes$ are associative binary operators over S with neutral elements $\mathbf{0}$ and $\mathbf{1}$, respectively, and such that $\oplus$ is commutative, and $\otimes$ distributes over $\oplus$ [10]. In the context of this paper, we consider two specific well known semirings: the *language* semiring, and the *trio* semiring.

The *language semiring* $\mathbb{L} = (\mathcal{L}(\Sigma), \cup, \cdot, \emptyset, \{\varepsilon\})$ is the semiring of all languages (that is, sets of finite words) over the alphabet Σ with the usual concatenation of languages $\cdot$, and the union of sets $\cup$. The empty word is denoted by ε. To reduce notation, we often represent singleton languages merely by the word they contain, when it is clear from the context.

The *trio semiring* $\mathbb{K} = (\mathbb{N}[\mathsf{N_V}], +, \times, 0, 1)$ is the semiring of polynomials with coefficients in $\mathbb{N}$ and variables in a countably infinite set $\mathsf{N_V}$, with the operation $+$ defined as usual and $\times$ idempotent and commutative [7,11,12]. We also consider in Section 6 the case in which $\times$ is non-commutative. Polynomials in the trio semiring in this work are in *expanded form*, meaning that they are sums of monomials. Every polynomial can be represented in this form.

Objects of the language and the trio semirings are similar, but have subtle differences: every language $\mathcal{L}$ can be seen as a (potentially infinite) polynomial, where each monomial is a word in $\mathcal{L}$. Conversely, the class of all monomials in a polynomial P in expanded form can be seen as a language modulo the commutativity of the product $\times$.

We consider a syntactic restriction of the ontology language $\mathcal{ELH}^r$ [1]. Concept and role names are taken from the disjoint countable sets $\mathsf{N_C}$ and $\mathsf{N_R}$, respectively, also disjoint from $\mathsf{N_V}$. $\mathcal{ELH}^r$ *general concept inclusions* (GCIs) $C \sqsubseteq D$ are built through the grammar rules $C ::= A \mid \exists R.C \mid C \sqcap C \mid \top$, $D ::= A \mid \exists R$, where $R \in \mathsf{N_R}$, $A \in \mathsf{N_C}$. *Role inclusions* (RIs) and *range restrictions* (RRs) are of the form $R \sqsubseteq S$ and $\mathsf{ran}(R) \sqsubseteq A$, respectively, with $R, S \in \mathsf{N_R}$ and $A \in \mathsf{N_C}$. An $\mathcal{ELH}^r$ *axiom* is a GCI, RI, or RR. An $\mathcal{ELH}^r$ TBox is a finite set of $\mathcal{ELH}^r$ axioms.

The reason for syntactically restricting $\mathcal{ELH}^r$ is that conjunctions or qualified restrictions of a role on the right-hand side of GCIs lead to counter-intuitive behavior when adding provenance annotations; see [5] for a detailed discussion on this issue.

An *annotated* $\mathcal{ELH}^r$ *TBox* $\mathcal{T}$ is a set of $\mathcal{ELH}^r$ axioms, each annotated with an element from $v \in \mathsf{N_V} \cup \{1\}$ representing provenance information. Axioms annotated with provenance information can be derived from an annotated ontology. They are annotated with monomials (potentially with more than one variable) representing the derivation of the axiom w.r.t. $\mathcal{O}$. From now on, $\mathsf{N_M}$ represents the set of all monomials.

An *annotated interpretation* is a triple $\mathcal{I} = (\Delta^\mathcal{I}, \Delta^\mathcal{I}_\mathsf{m}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}, \Delta^\mathcal{I}_\mathsf{m}$ are non-empty disjoint sets (the *domain* and *domain of monomials* of $\mathcal{I}$, respectively), and $\cdot^\mathcal{I}$ maps

- every $A \in \mathsf{N_C}$ to $A^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}$;
- every $R \in \mathsf{N_R}$ to $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \times \Delta^\mathcal{I}_\mathsf{m}$; and
- every $m, n \in \mathsf{N_M}$ to $m^\mathcal{I}, n^\mathcal{I} \in \Delta^\mathcal{I}_\mathsf{m}$ s.t. $m^\mathcal{I} = n^\mathcal{I}$ iff m and n are equal modulo associativity, commutativity and ×-idempotency (e.g., $(n \times m)^\mathcal{I} = (m \times n)^\mathcal{I}$).

As mentioned, we consider in Section 6 the case in which × is non-commutative. We extend $\cdot^\mathcal{I}$ to complex $\mathcal{ELH}^r$ expressions as usual:

$$\begin{aligned}
(\top)^\mathcal{I} &= \Delta^\mathcal{I} \times \{1^\mathcal{I}\};\\
(\exists R)^\mathcal{I} &= \{(d, m^\mathcal{I}) \mid \exists e \in \Delta^\mathcal{I} \text{ s.t. } (d, e, m^\mathcal{I}) \in R^\mathcal{I}\};\\
(C \sqcap D)^\mathcal{I} &= \{(d, (m \times n)^\mathcal{I}) \mid (d, m^\mathcal{I}) \in C^\mathcal{I}, (d, n^\mathcal{I}) \in D^\mathcal{I}\};\\
(\mathsf{ran}(R))^\mathcal{I} &= \{(e, m^\mathcal{I}) \mid \exists d \in \Delta^\mathcal{I} \text{ s.t. } (d, e, m^\mathcal{I}) \in R^\mathcal{I}\};\\
(\exists R.C)^\mathcal{I} &= \{(d, (m \times n)^\mathcal{I}) \mid \exists e \in \Delta^\mathcal{I} \text{ s.t.}\\
&\qquad (d, e, m^\mathcal{I}) \in R^\mathcal{I}, (e, n^\mathcal{I}) \in C^\mathcal{I}\}.
\end{aligned}$$

The annotated interpretation $\mathcal{I}$ *satisfies*: $(R \sqsubseteq S, m)$ if for all $n \in \mathsf{N_M}$, $(d, e, n^\mathcal{I}) \in R^\mathcal{I}$ implies $(d, e, (m \times n)^\mathcal{I}) \in S^\mathcal{I}$; and $(C \sqsubseteq D, m)$ if for all $n \in \mathsf{N_M}$, $(d, n^\mathcal{I}) \in C^\mathcal{I}$ implies $(d, (m \times n)^\mathcal{I}) \in D^\mathcal{I}$. $\mathcal{I}$ is a *model* of $\mathcal{T}$, denoted $\mathcal{I} \models \mathcal{T}$, if it satisfies all annotated axioms in $\mathcal{T}$. $\mathcal{T}$ *entails* (α, m), denoted $\mathcal{O} \models (\alpha, m)$, if $\mathcal{I} \models (\alpha, m)$ for every model $\mathcal{I}$ of $\mathcal{O}$.

We are interested in the provenance for a subsumption problem: given a TBox $\mathcal{T}$ and two concept names A, B, find all monomials m such that $\mathcal{T} \models (A \sqsubseteq B, m)$. We solve it by constructing an ARA that accepts representatives of all these monomials. We use this construction to answer, given a monomial m, whether $\mathcal{T} \models (A \sqsubseteq B, m)$ holds, and show that this problem is in NP.

3 Automata

We consider two generalisations of non-deterministic finite automata (NFA) [13]; namely, weighted tree automata and acyclic recursive automata.

3.1 Weighted Tree Automata

Tree automata [8] generalise NFAs by accepting trees rather than words; the branching of the trees is identified by the *arity* of the automaton. Weighted tree automata further generalise this notion by not only accepting or rejecting an input tree, but assigning a value from a given semiring (its *weight*) [16]. For the scope of this paper, we consider only *unlabelled* trees.

Let $k \geq 1$. A *weighted tree automaton* over $\mathbb{S}$ of arity k is a tuple of the form $\mathcal{A} = (Q, \mathbb{S}, \mathsf{wt}, I, f)$ where Q is a finite set of *states*, $\mathbb{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is a semiring, $\mathsf{wt} : Q^{k+1} \to S$ is the *transition weight function*, $I \subseteq Q$ is the set of *initial states*, and $f : Q \to S$ is the *exit weight function*.

As usual, we represent trees of arity k as finite non-empty sets $T \subseteq \{1, \dots, k\}^*$ such that if $wi \in T$, then $w, wj \in T$ for each $w \in \{1, \dots, k\}^*, 1 \leq i, j \leq k$. Given a tree T of arity k, a *run* $\rho : T \to Q$ of $\mathcal{A}$ over T assigns a state to each node in T. The *weight* of this run is $\mathsf{wt}(\rho) = \bigotimes_{w1 \in T} \mathsf{wt}(\rho(w), \rho(w1), \dots, \rho(wk)) \otimes \bigotimes_{w1 \notin T} f(\rho(w))$; that is, the product of all the transition and exit weights given the states assigned by ρ. For non-commutative semirings, this product is made from the root to the leafs, and in the order of the children (i.e., top-down, left-to-right). Given a state $q \in Q$, we define $\mathsf{wt}(q) = \bigoplus_{\rho(\varepsilon)=q} \mathsf{wt}(\rho)$; that is, the sum of the weights of all runs that label the root of a tree with q. The *behaviour* of the automaton $\mathcal{A}$ is the sum of the weights of all its initial states $\|\mathcal{A}\| := \bigoplus_{q \in I} \mathsf{wt}(q)$.

3.2 Acyclic Recursive Automata

Acyclic recursive automata generalise NFA by allowing an automaton to call another one, but the calls between automata must respect a hierarchical ordering. They were originally introduced as *hierarchical state machines* [20] with a slightly different structure.

Definition 1 (ARA). *An* acyclic recursive automaton *over the alphabet Σ is a finite set $\mathfrak{A} = \{\mathcal{A}_i \mid i \in I\}$ of NFAs $\mathcal{A}_i = (Q_i, \Sigma_i, \Delta_i, I_i, F_i)$, where $(I, \leq)$ is a partially ordered set of indices, such that: (i) for all $i \neq j \in I$, $Q_i \cap Q_j = \emptyset$; (ii) $\Sigma_i = \Sigma \cup \{\mathsf{m}_j \mid j < i\}$; and (iii) $\{\mathsf{m}_i \mid i \in I\} \cap \Sigma = \emptyset$.*

We call the symbols m_i, which are added to the alphabets of the different automata in $\mathfrak{A}$, *call triggers*. When an automaton $\mathcal{A}_i$ reads the symbol m_j, it "calls" the automaton $\mathcal{A}_j$, which continues reading the word until it chooses to return the control to the "calling" automaton $\mathcal{A}_i$ (signalled by the symbol $\overline{\mathsf{m}}_j$). This return is only possible if A_j is in one of its accepting states. In practice, the automaton $\mathcal{A}_j$ is in charge of accepting a portion of the input word.

Each automaton $\mathcal{A}_i$ may call any other automaton $\mathcal{A}_j$ where $j < i$. Hence, there may be a sequence of nested calls, but the depth of this nesting is always bounded by the number n of automata in $\mathfrak{A}$. Moreover, the automaton $\mathcal{A}_i$ can never call itself either directly or indirectly. To define the language accepted by the ARA $\mathfrak{A}$, we adapt the notion of a run to take into account also the nested calls between the automata.

Definition 2 (valid run). *A* run *of the ARA $\mathfrak{A} = \{\mathcal{A}_j \mid j \in I\}$ is a finite sequence $\rho = q_0, s_1, q_1, \ldots, s_k, q_k$ such that $q_i \in \bigcup_{j \in I} Q_j$ for all $0 \leq i \leq k$ and $s_i \in \Sigma \cup \{\mathsf{m}_j, \overline{\mathsf{m}}_j \mid j < i\}$ for all $1 \leq i \leq k$. The notion of a valid run on an automaton $\mathcal{A}_i$ is inductively defined as follows. The run $\rho = q_0, s_1, q_1, \ldots, s_k, q_k$ is* valid *on $\mathcal{A}_i$ iff*

- *$\{s_1, \ldots, s_k\} \subseteq \Sigma$ and $(q_j, s_{j+1}, q_{j+i}) \in \Delta_i$ for all $0 \leq j < k$ or*
- *j is the smallest index such that $s_j \notin \Sigma$, s_j is of the form m_ℓ, there exists $j' > j$ with $s_{j'} = \overline{\mathsf{m}}_\ell$, and*
  - *$q_0, s_1, q_2, \ldots, s_{j-1}, q_{j-1}$ and $q_{j'}, s_{j'+1}, \ldots, q_k$ are valid in $\mathcal{A}_i$*
  - *$q_j, s_{j+1}, q_{j+1}, \ldots, s_{j'-1}, q_{j'-1}$ is valid in $\mathcal{A}_\ell$ and*
  - *$(q_{j-1}, \mathsf{m}_\ell, q_{j'}) \in \mathcal{A}_i$*

So far, we have not yet expressed the use of initial and final states in accepting a word. In a nutshell, whenever we call an automaton, its execution should accept a segment of the input word, by traversing from an initial to a final state.

Definition 3 (successful run). *Given a valid run $\rho = q_0, s_1, q_1, \ldots, s_k, q_k$, the index $i, 1 \leq i \leq k$ is called a* top-level call index *iff $s_i = \mathsf{m}_j$ for some j, and for every $\ell < i$ such that $s_\ell = \mathsf{m}_{j'}$ there is an $\ell', \ell < \ell' < i$ such that $s_{\ell'} = \overline{\mathsf{m}}_{j'}$. If i is a top-level call index with $s_i = \mathsf{m}_j$, then the smallest index $\ell, i < \ell \leq k$ such that $s_\ell = \overline{\mathsf{m}}_j$ is its* match*. This is denoted as $\ell = \mathsf{match}(i)$.*

A valid run $\rho = q_0, s_1, q_1, \ldots, s_k, q_k$ is successful *in $\mathcal{A}_i$ iff $q_0 \in I_i$, $q_k \in F_i$ and for every top-level call index i with $s_i = \mathsf{m}_j$ and $\ell = \mathsf{match}(i)$, the sequence $q_i, s_{i+1}, \ldots, q_{\ell-1}$ is a successful run in $\mathcal{A}_j$.*

The run ρ is successful in the ARA $\mathfrak{A} = \{\mathcal{A}_i \mid i \in I\}$ iff it is successful in $\mathcal{A}_j$, for some maximal element j of I. The word accepted *by this run is the concatenation of all symbols of Σ appearing in ρ. The* language *of $\mathfrak{A}$ is the set $\mathcal{L}(\mathfrak{A})$ of all words accepted by a successful run in $\mathfrak{A}$. By extension, the language accepted by $\mathcal{A}_i$ is the set $\mathcal{L}(\mathcal{A}_i)$ of all words accepted by a successful run in $\mathcal{A}_i$, for each $i \in I$.*

ARAs are not more expressive than NFAs; they also accept regular languages. The main difference is that an ARA can be exponentially more succinct than an NFA for representing a given language. For example, the language $\{a^{2^n}\}$ that contains only one word with 2^n symbols a can only be recognised by NFAs with at least 2^n states, but is accepted by an ARA having n automata with 3 states each (hence $3n$ states in total); see the appendix of the extended version [17]. The *size* of the ARA $\mathfrak{A}$ is the total number of states in the NFAs in $\mathfrak{A}$.

The relevant properties of ARAs for this paper are the following. Deciding whether the ARA $\mathfrak{A}$ accepts a word w requires only polynomial time. The concatenation of n ARAs is obtained by adding a new NFA with $n+1$ states that calls each ARA once. The union of n ARAs is obtained by adding a new NFA with 2 states, which non-deterministically calls one of the ARAs. Abusing the notation, given two ARAs $\mathfrak{A}$, $\mathfrak{B}$ we denote as $\mathfrak{A} \cdot \mathfrak{B}$ and $\mathfrak{A} \cup \mathfrak{B}$ the ARAs obtained through these constructions, respectively.

Table 1. Transitions of $\mathcal{A}_{A_0 \sqsubseteq B_0}$ with weight $\{\varepsilon\}$.

$$\begin{aligned}
T = \{&(R_1 \sqsubseteq R_3, R_1 \sqsubseteq R_2, R_2 \sqsubseteq R_3, \square, \square, \square),\\
&(\mathsf{ran}(R) \sqsubseteq A, R \sqsubseteq S, \mathsf{ran}(S) \sqsubseteq A, \square, \square, \square),\\
&(A \sqsubseteq \exists S, A \sqsubseteq \exists R, R \sqsubseteq S, \square, \square, \square), (A \sqsubseteq C, A \sqsubseteq B, B \sqsubseteq C, \square, \square, \square),\\
&(A \sqsubseteq \exists R, A \sqsubseteq B, B \sqsubseteq \exists R, \square, \square, \square),\\
&(A \sqsubseteq C, A \sqsubseteq B_1, A \sqsubseteq B_2, B_1 \sqcap B_2 \sqsubseteq C, \square, \square),\\
&(\mathsf{ran}(R) \sqsubseteq C, \mathsf{ran}(R) \sqsubseteq B_1, \mathsf{ran}(R) \sqsubseteq B_2, B_1 \sqsubseteq C_1, B_2 \sqsubseteq C_2, C_1 \sqcap C_2 \sqsubseteq C),\\
&(A \sqsubseteq C, A \sqcap B \sqsubseteq C, \top \sqsubseteq B, \square, \square, \square),\\
&(A \sqsubseteq D, A \sqsubseteq \exists S, \mathsf{ran}(S) \sqsubseteq B, B \sqsubseteq C, S \sqsubseteq R, \exists R.C \sqsubseteq D),\\
&(A \sqsubseteq C, A \sqsubseteq \exists R, \top \sqsubseteq B, \exists R.B \sqsubseteq C, \square, \square)\\
&\qquad \mid A, B, C, D \in N_C(\mathcal{T}) \cup \{\top\}, R, S \in N_R(\mathcal{T})\}
\end{aligned}$$

4 The Weighted Automaton

Our goal is to build an ARA which accepts representatives for all the monomials in the provenance of a subsumption relation. To do so, we first present a weighted tree automaton whose behaviour (which is a language) can be seen as a polynomial (in expanded form) constructed by the provenance monomials for the desired consequence; modulo commutativity. The method for *computing* this behaviour will give rise to the ARA.

The construction of the automaton is based on considering the "proofs" of a derivation based on the consequence-based algorithm, built in a top-down manner (from the desired consequence, deconstructed back to the axioms used). Formally, we have a different automaton for each consequence that we might want to verify. However, all the automata are equivalent, except for the initial state; which refers to the desired consequence. The automaton, which reads trees of arity 5, is also very simple because all transitions that refer to a consequence step have weight $\{\varepsilon\}$ (the neutral of the language semiring product), and the only "real" weight is found at the final states (the exit weight) which is given by the provenance label of the axiom in the TBox.

Let A_0, B_0 be two distinguished concept names appearing in an annotated TBox $\mathcal{T}$; the weighted automaton $\mathcal{A}_{A_0 \sqsubseteq B_0} = (Q \cup \{\square\}, \mathbb{L}, \mathsf{wt}, I, f)$ is given by

- Q is the set of all axioms in restricted normal form on the alphabet of $\mathcal{T}$;
- $\mathsf{wt}(\delta) = \{\varepsilon\}$ if $\delta \in T$ (see Table 1) and $\mathsf{wt}(\delta) = \emptyset$ otherwise;
- $I = \{A_0 \sqsubseteq B_0\}$;
- $f(q) = \{v\}$ if $(q, v) \in \mathcal{T}$; $f(q) = \{\varepsilon\}$ if $q \in \{X \sqsubseteq X, X \sqsubseteq \top, \square\}$; and $f(q) = \emptyset$ otherwise.

The special symbol $\square$ is used to keep the arity of the automaton to 5. In a nutshell, the transitions of this automaton can be seen as the completion rules from [5], but applied *backwards*, from the consequence to the premises that generate it. The weight of any run which labels the root with $A_0 \sqsubseteq B_0$ is either

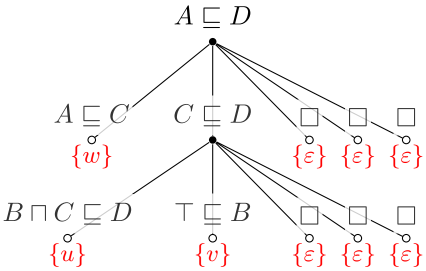

Fig. 1. One run of the tree automaton $\mathcal{A}_{A \sqsubseteq D}$ from Example 4. Its weight is $\{wuv\}$.

$\emptyset$ if the labelled tree does not represent a derivation of the consequence, or a single word concatenating the annotations of the axioms from $\mathcal{T}$ used in the derivation. Modulo idempotency, this word represents a provenance monomial for $A_0 \sqsubseteq B_0$. The behaviour of the automaton is then the language containing a representation of all such provenance monomials.

Example 4. Consider the annotated TBox $\mathcal{T}$ containing the following five axioms $\mathcal{T} := \{(B \sqcap C \sqsubseteq D, u), (\top \sqsubseteq B, v), (A \sqsubseteq C, w), (A \sqsubseteq \exists R, x), (\exists R.B \sqsubseteq B, y)\}$. One possible run of the automaton $\mathcal{A}_{A \sqsubseteq D}$ is depicted in Figure 1. The two internal nodes (marked with $\bullet$) have a transition weight of $\{\varepsilon\}$. The weight of this run is $\{wuv\}$. It can be seen that every non-leaf node is the consequence obtained from its successors (ignoring the dummy nodes $\square$). There are at least two other runs with weight different from $\emptyset$; one has weight $\{vwu\}$ and the other $\{xvywu\}$. The behaviour $\|\mathcal{A}_{A \sqsubseteq D}\|$ contains $\{wuv, vwu, xvywu\}$. Note that the first two words correspond to the same provenance monomial in $\mathbb{K}$, due to commutativity. In $\mathbb{L}$, they are two different objects.

As it can be seen from the example, the behaviour of $\mathcal{A}_{A_0 \sqsubseteq B_0}$ does not directly yield the set of all provenance monomials for $A_0 \sqsubseteq B_0$. However, these monomials can be extracted from $\|\mathcal{A}_{A_0 \sqsubseteq B_0}\|$ by taking into account the commutativity and idempotency of $\mathbb{K}$. Abusing the notation, given a word ω, we will denote as $[\omega]$ a representative monomial ω w.r.t. commutativity and idempotency. Hence for instance $[xuxv] = uvx$. The next theorem is a direct consequence of the correctness of the completion algorithm [5].

Theorem 5. *There is a run ρ of $\mathcal{A}_{A_0 \sqsubseteq B_0}$ with weight $\mathsf{wt}(\rho) = \{m\} \neq \{\varepsilon\}$ iff $\mathcal{T} \models (A_0 \sqsubseteq B_0, [m])$.*

Thus, the behaviour of this automaton, which accumulates the weights of all possible runs, represents all provenance monomials for the consequence $A_0 \sqsubseteq B_0$. The question is: how to find this behaviour? Answering this question is the scope of the following section; but before that, we emphasise that the automata for different consequences are all identical except for the initial state, which is used to label the root node (that is, the goal that we aim to reach through a proof). Hence, for the TBox in Example 4, we get $\{v, xvy\} \subseteq \|\mathcal{A}_{A \sqsubseteq B}\|$.

5 The Behaviour

Following the general idea from [2, 9], we compute the behaviour of the automaton via a bottom-up approach, by iteratively accumulating the provenance of intermediate consequences used in the derivation of $A_0 \sqsubseteq B_0$. However, the technique must be adapted to handle the semiring $\mathbb{L}$, which is not a lattice.

Specifically, we build the functions $\mathsf{wt}_i : Q \to \mathcal{L}(N_V)$, $i \in \mathbb{N}$ as follows:

- $\mathsf{wt}_0 = f$;[1]
- for $i \geq 0$, $\mathsf{wt}_{i+1}(q) = \mathsf{wt}_i(q) \cup \bigcup_{(q,q_1,\dots,q_5)\in T} \mathsf{wt}_i(q_1) \cdot \cdots \cdot \mathsf{wt}_i(q_5)$

It can be shown by induction on i that $\mathsf{wt}_i(q)$ has a representative for all the monomials arising from trees with root labelled with q and depth at most i. In particular for $i = 0$, $\mathsf{wt}_0(q)$ is the label of the axiom q if it appears in $\mathcal{T}$, $\{\varepsilon\}$ if q is a tautology or $\square$, and $\emptyset$ otherwise. Importantly, $\mathsf{wt}_i(q) \subseteq \mathsf{wt}_{i+1}(q)$ for all $q \in Q$ and all $i \in \mathbb{N}$. We can thus see the construction of wt_i as a monotone operator which, in particular, has a smallest fixpoint: the limit of the functions wt_i. This fixpoint is, in fact, the behaviour of $\mathcal{A}_{A_0 \sqsubseteq B_0}$.

Theorem 6. *The behaviour of $\mathcal{A}_{A_0 \sqsubseteq B_0}$ is $\lim_{n\to\infty} \mathsf{wt}_n(A_0 \sqsubseteq B_0)$.*

Importantly, the functions wt_i actually assign a language to each state of the automaton. To find out the behaviour of a different consequence, say $A_1 \sqsubseteq B_1$, one does not need to recompute the automaton and the functions wt_i, but needs to find $\lim_{n\to\infty} \mathsf{wt}_n(A_1 \sqsubseteq B_1)$. In other words, finding these functions provides enough information for computing the provenance monomials of all possible consequences (in normal form) from the TBox.

In general, the construction of wt_i will not yield the fixpoint after finitely many applications. Indeed, w.r.t. TBox $\{(A \sqsubseteq B, u), (B \sqsubseteq A, v)\}$, we get that $\lim_{n\to\infty} \mathsf{wt}_n(A \sqsubseteq B) = (uv)^* u$, but each $\mathsf{wt}_i(A \sqsubseteq B)$ contains finitely many words. However, recall that we are not interested in the language $\|\mathcal{A}_{A_0 \sqsubseteq B_0}\|$ *per se*, but rather in the monomials that the words in this language represent. Since the Trio semiring (which we use to characterise the provenance) uses a commutative and idempotent product operation, we are only interested in the *symbols* that appear in the words, and not in the actual words themselves. That is, we are only interested in the languages up to representative monomials.

Definition 7 ($\mathbb{K}$-equivalence). *Two languages $\mathcal{L}, \mathcal{L}'$ are* $\mathbb{K}$-equivalent *(denoted as $\mathcal{L} \equiv_{\mathbb{K}} \mathcal{L}'$) iff $\{[\omega] \mid \omega \in \mathcal{L}\} = \{[\omega] \mid \omega \in \mathcal{L}'\}$.*

For example, $(uv)^* u$ and $\{uv, u\}$ are $\mathbb{K}$-equivalent. While the languages $\mathsf{wt}_i(q)$ and the words therein may grow indefinitely, their representative monomials are limited by the provenance variables appearing in $\mathcal{T}$; which are at most $|\mathcal{T}|$. Hence, there exists an $n \in \mathbb{N}$ such that $\mathsf{wt}_m(q) \equiv_{\mathbb{K}} \mathsf{wt}_n(q)$ holds for all $q \in Q$ and all $m \geq n$. Following Theorem 5, for this n $\mathsf{wt}_n(A_0 \sqsubseteq B_0)$ contains representatives for all the provenance monomials for $A_0 \sqsubseteq B_0$.

[1] Recall that f is the exit weight function of the automaton.

Table 2. Extensional description of the languages wt_i for the TBox from Example 4.

	$\square$	$B \sqcap C \sqsubseteq D$	$\top \sqsubseteq B$	$A \sqsubseteq C$	$A \sqsubseteq \exists R$	$\exists R.B \sqsubseteq B$	$A \sqsubseteq \top$	$A \sqsubseteq B$	$C \sqsubseteq D$	$A \sqsubseteq D$
wt_0	$\{\varepsilon\}$	$\{u\}$	$\{v\}$	$\{w\}$	$\{x\}$	$\{y\}$	$\{\varepsilon\}$	$\emptyset$	$\emptyset$	$\emptyset$
wt_1	$\{\varepsilon\}$	$\{u\}$	$\{v\}$	$\{w\}$	$\{x\}$	$\{y\}$	$\{\varepsilon\}$	$\{v, xvy\}$	$\{uv\}$	$\emptyset$
wt_2	$\{\varepsilon\}$	$\{u\}$	$\{v\}$	$\{w\}$	$\{x\}$	$\{y\}$	$\{\varepsilon\}$	$\{v, xvy\}$	$\{uv\}$	$\{wuv, vwu, xvywu\}$
wt_3	$\{\varepsilon\}$	$\{u\}$	$\{v\}$	$\{w\}$	$\{x\}$	$\{y\}$	$\{\varepsilon\}$	$\{v, xvy\}$	$\{uv\}$	$\{wuv, vwu, xvywu\}$

As argued before, $\mathsf{wt}_i(q)$ contains the weights of all runs of height at most i with root q. It can be seen that for every run ρ of height greater than $|Q| \cdot |\mathcal{T}|$ there is a smaller run ρ' such that $\mathsf{wt}(\rho) = \mathsf{wt}(\rho')$. This means that the least fixpoint for wt_i is found after at most $|Q| \cdot |\mathcal{T}|$ iterations, which is polynomial in $|\mathcal{T}|$. Specifically, the number of iterations needed to reach a fixpoint is bounded by $\mathcal{O}(|\mathcal{T}|^4)$.

Recall that each $\mathsf{wt}_i(q)$ is a language. By construction, it is a regular language; indeed, it is formed by concatenation and union of finite languages. If we tried to represent these languages *extensionally*, enumerating all the words they contain, we would potentially need exponential space: potentially, the language may contain exponentially many words. Exploiting the fact that these languages are regular, we can represent them through NFAs. In fact, wt_0 is composed of very simple automata with at most two states, and the construction of wt_{i+1} from wt_i requires only concatenation and union of automata, which are basic automata operations [13]. However, iteratively constructing these NFA as in the definition of wt_i can also lead to an exponential blowup; for an example see the extended version of this paper [17]. To keep the construction tractable, we exploit the succinctness power of ARAs.

Note once again that each $\mathsf{wt}_0(q)$ contains either a word of length 1, the empty word, or is the empty language. All these languages are recognisable by NFA with at most two states. We call these automata $\mathcal{A}_0^q$. For each successive $\mathsf{wt}_{i+1}(q)$ we construct an automaton $\mathcal{A}_{i+1}^q$ that calls the automata $\mathcal{A}_i^{q'}$, which accept the languages $\mathsf{wt}_i(q'), q' \in Q$. Thus we are constructing an ARA with the ordering $\mathcal{A}_i^q \leq \mathcal{A}_j^{q'}$ for all $q, q' \in Q$ and all $0 \leq i < j$. Importantly, each automaton $\mathcal{A}_{i+1}^q$ requires at most five states (to concatenate the languages of the successive states) for each transition $(q, q_1, \dots, q_5) \in T$ (recall Table 1). Since the number of such transitions is bounded by $|Q|^5$, it follows that the size of each ARA $\mathfrak{A}_i^q := \{\mathcal{A}_j^{q'} \mid q' \in Q, j < i\} \cup \{\mathcal{A}_i^q\}$ is in $\mathcal{O}(|Q|^5 \cdot i)$. Let now $\mathfrak{A}^q := \mathfrak{A}_n^q$, where n is the number of iterations needed to reach a fixpoint w.r.t. $\mathbb{K}$-equivalence. As seen, its size is in $\mathcal{O}(|Q|^6 \cdot |\mathcal{T}|)$; that is, it is bounded by a polynomial on $|\mathcal{T}|$. Moreover, this ARA $\mathfrak{A}^q$ suffices to find all the provenance monomials for the consequence q, as expressed next.

Theorem 8. *$\mathcal{T} \models (A_0 \sqsubseteq B_0, m)$ iff there is a word $\omega \in \mathcal{L}(\mathfrak{A}^{A_0 \sqsubseteq B_0})$ such that $[m] = [\omega]$.*

Example 9. Consider again the TBox from Example 4. The languages wt_i are extensionally represented in Table 2. The construction of the automata A_{i+1}^q

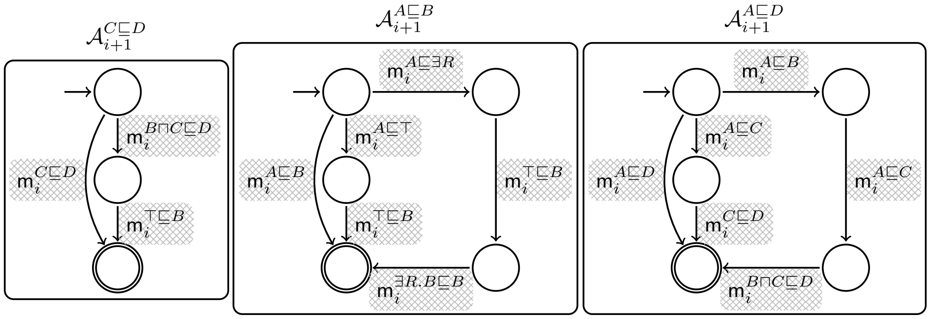
$\mathcal{A}_{i+1}^{C\sqsubseteq D}$
$\mathsf{m}_i^{B\sqcap C\sqsubseteq D}$
$\mathsf{m}_i^{C\sqsubseteq D}$
$\mathsf{m}_i^{\top\sqsubseteq B}$
$\mathcal{A}_{i+1}^{A\sqsubseteq B}$
$\mathsf{m}_i^{A\sqsubseteq\exists R}$
$\mathsf{m}_i^{A\sqsubseteq\top}$
$\mathsf{m}_i^{A\sqsubseteq B}$
$\mathsf{m}_i^{\top\sqsubseteq B}$
$\mathsf{m}_i^{\top\sqsubseteq B}$
$\mathsf{m}_i^{\exists R.B\sqsubseteq B}$
$\mathcal{A}_{i+1}^{A\sqsubseteq D}$
$\mathsf{m}_i^{A\sqsubseteq B}$
$\mathsf{m}_i^{A\sqsubseteq C}$
$\mathsf{m}_i^{A\sqsubseteq D}$
$\mathsf{m}_i^{A\sqsubseteq C}$
$\mathsf{m}_i^{C\sqsubseteq D}$
$\mathsf{m}_i^{B\sqcap C\sqsubseteq D}$

Fig. 2. Automata for computing $\|\mathcal{A}_{A\sqsubseteq D}\|$ w.r.t. the TBox in Example 4.

for $i \geq 0$ and $q \in \{C \sqsubseteq D, A \sqsubseteq B, A \sqsubseteq D\}$ is depicted in Figure 2, where each transition m_i^q is a call to the automaton $\mathcal{A}_i^q$. Hence, for instance $\mathcal{A}_1^{C\sqsubseteq D}$ may non-deterministically call $\mathcal{A}_0^{C\sqsubseteq D}$ (which yields the empty language), or concatenate a word accepted by $\mathcal{A}_0^{B\sqcap C\sqsubseteq D}$ with a word from $\mathcal{A}_0^{\top\sqsubseteq B}$. Thus, $\mathcal{L}(\mathcal{A}_1^{C\sqsubseteq D}) = \{uv\}$. Similarly, we can see that $\mathcal{L}(\mathcal{A}_1^{A\sqsubseteq D}) = \emptyset$. Note that for a fixed $q \in Q$ the structure of the automata $\mathcal{A}_{i+1}^q$ is the same for all $i \geq 0$. The difference is that they call the automata from the previous iteration.

Recall that deciding whether a word ω is a accepted by an ARA $\mathfrak{A}$ is polynomial on the size of $\mathfrak{A}$. In particular, for $\mathfrak{A}^q$ this task is polynomial on $|\mathcal{T}|$. However, Theorem 8 requires to first find the word ω that needs to be tested. One idea is to build an automaton $\mathcal{A}$ that accepts the language $\mathcal{L}_m := \{\omega \mid [\omega] = [m]\}$ and check whether $\mathcal{L}(\mathfrak{A}^q) \cap \mathcal{L}_m \neq \emptyset$. However, it is not at all clear whether $\mathcal{L}_m$ is even a regular language; specifically, to the best of our knowledge it has never been verified whether the commutative closure of a regular language it also regular.

To solve this issue, we first *guess* (in polynomial time on the size of m) an ordering of the symbols in m—say $\sigma_1, \dots, \sigma_k$—and then verify whether $\mathfrak{A}^q$ accepts a word from

$$\sigma_1^+\sigma_2(\sigma_1 \cup \sigma_2)^*\sigma_3(\sigma_1 \cup \sigma_2 \cup \sigma_3)^*\sigma_4 \cdots (\bigcup_{i=1}^{k-1} \sigma_i)^*\sigma_k(\bigcup_{i=1}^{k} \sigma_i)^*; \tag{1}$$

that is, a word where the symbols first appear in the specified order. Note that the language in Equation (1) is regular, and can be recognised by an NFA with $k+1$ states. Recall also that given an ARA $\mathfrak{A}$ and an NFA $\mathcal{A}$, it is possible to construct an ARA $\mathfrak{A}'$ of size bounded by $|\mathfrak{A}||\mathcal{A}|$ such that $\mathcal{L}(\mathfrak{A}') = \mathcal{L}(\mathfrak{A}) \cap \mathcal{L}(\mathcal{A})$. Thus, verifying whether the chosen order yields a word accepted by $\mathfrak{A}^q$ is polynomial on $|\mathcal{T}|$ and $|m|$. The non-deterministic ordering guess yields the following.

Theorem 10. *Deciding* $\mathcal{T} \models (A_0 \sqsubseteq B_0, m)$ *is in* NP.

6 The Non-commutative Case

We now consider the case where the semiring is not commutative. The idea of non-commutativity is to preserve the information of the order in which axioms were used to derive a consequence. We consider here a *left-absorbing product*: for multiple occurrences of the same provenance symbol, we take into account the first (or left-most) one. Thus, e.g., $[uvu] = [uv] \neq [vu]$. We call this case *non-commutative* $\mathcal{ELH}^r$.

Example 11. Let $\mathcal{T} := \{(A \sqsubseteq B, m), (B \sqsubseteq C, n)\}$. In non-commutative $\mathcal{ELH}^r$, $\mathcal{T} \models (A \sqsubseteq C, mn)$ but $\mathcal{T} \not\models (A \sqsubseteq C, nm)$.

Non-commutativity also means that, e.g., the concept expression $A \sqcap B$ is not interpreted in the same way as $B \sqcap A$, which may seem counterintuitive since in classical DL semantics these concepts are equivalent. One possible use case for this semantics is for representing definitional sentences in natural language processing [15, 18], where the order of the words usually also changes the meaning. For example, the logic would distinguish $\mathsf{White} \sqcap \mathsf{Wine}$ from $\mathsf{Wine} \sqcap \mathsf{White}$.

Interestingly, we know from Equation (1) that we can verify whether $\mathfrak{A}^q$ accepts a representative (under left-absorption) of a monomial m. The benefit in this case is that it is not necessary to first guess the right ordering, as it is required by the ordering given in m. This yields the following result.

Theorem 12. *$\mathcal{T} \models (A_0 \sqsubseteq B_0, m)$ w.r.t. a left-absorbing, non-commutative semiring can be decided in polynomial time.*

7 Conclusions

In this paper we have studied the complexity of deciding whether the provenance of a subsumption relation contains a given monomial m. In previous work [5], it was shown through a completion algorithm, that this problem is in PSpace when the semiring product is idempotent and commutative, but only a polynomial lower bound (derived from reasoning in $\mathcal{ELH}^r$) was given. By viewing the completion algorithm backwards, as a decomposition approach based on tree automata, and exploiting a less known class of automata (ARAs) to simulate structure sharing, we were able to lower this upper bound to NP. Unfortunately, the polynomial lower bound remains the best available at the moment. If we substitute commutativity by a notion of left-absorption, we obtain a tight polynomial-time complexity for this problem. Interestingly, the technique developed can be applied to instance queries and assertion entailments, simply by extending the automaton construction to the ABox-handling rules from [5]. Hence, the same complexity bounds hold in both cases. One avenue for future work is to close the remaining complexity gaps in these problems.

Note that the complexity results depend strongly on the properties of the provenance semiring. Indeed, the fixpoint computation of the functions wt_i only terminates due to the idempotence and commutativity (or left-absorption) of the product. However, the construction of the automaton $\mathcal{A}_{A_0 \sqsubseteq B_0}$ remains correct for a larger class of semirings (Theorem 5). We will study whether the technique can be applied in practice for these other semirings. One particular point of interest is to consider *closure semirings* [14] or other approaches for handling the repeating structure of the ARAs.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
2. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. J. Autom. Reasoning **45**(2), 91–129 (2010). https://doi.org/10.1007/s10817-010-9181-2, https://doi.org/10.1007/s10817-010-9181-2
3. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. J. Log. Comput. **20**(1), 5–34 (2010). https://doi.org/10.1093/logcom/exn058, https://doi.org/10.1093/logcom/exn058
4. Bourgaux, C., Ozaki, A.: Querying attributed DL-Lite ontologies using provenance semirings. In: AAAI (2019)
5. Bourgaux, C., Ozaki, A., Peñaloza, R., Predoiu, L.: Provenance for the description logic elhr. In: Bessiere, C. (ed.) IJCAI. pp. 1862–1869. ijcai.org (2020)
6. Calvanese, D., Lanti, D., Ozaki, A., Peñaloza, R., Xiao, G.: Enriching ontology-based data access with provenance. In: IJCAI (2019)
7. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. Foundations and Trends in Databases **1**(4), 379–474 (2009)
8. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata (2007), release October, 12th 2007
9. Droste, M., Kuich, W., Rahonis, G.: Multi-valued mso logics overwords and trees. Fundam. Inf. **84**(3,4), 305–327 (Dec 2008)
10. Golan, J.S.: The theory of semirings with applications in mathematics and theoretical computer science, Pitman monographs and surveys in pure and applied mathematics, vol. 54. Longman Scientific & Technical (1992)
11. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: PODS (2007)
12. Green, T.J., Tannen, V.: The semiring framework for database provenance. In: PODS (2017)
13. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company (1979)
14. Lehmann, D.J.: Algebraic structures for transitive closure. Theoretical Computer Science **4**(1), 59–76 (1977). https://doi.org/https://doi.org/10.1016/0304-3975(77)90056-1
15. Ma, Y., Distel, F.: Learning formal definitions for Snomed CT from text. In: AIME. pp. 73–77 (2013)
16. M.Droste, W.Kuich, H.Vogler (eds.): Handbook of Weighted Automata. Monographs in Theoretical Computer Science, Springer (2009)
17. Peñaloza, R.: An upper bound for provenance in $\mathcal{ELH}^r$. CoRR **abs/2108.12774** (2021)

18. Petrucci, G., Ghidini, C., Rospocher, M.: Ontology learning in the deep. In: EKAW. pp. 480–495 (2016)
19. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI (2003)
20. Yannakakis, M.: Hierarchical state machines. In: van Leeuwen, J., Watanabe, O., Hagiya, M., Mosses, P.D., Ito, T. (eds.) Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics. pp. 315–330. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)