

Tableau Algorithm for Concept Satisfiability in Description Logic \mathcal{ALCH}

Satya S. Sahoo^{1*}, Krishnaprasad Thirunarayan^{1*}

¹ Kno.e.sis Center, Computer Science and Engineering Department, Wright State University,
Dayton, OH-45435, USA
{sahoo.2, t.k.prasad}@wright.edu

Abstract. The *provenir* ontology is an upper-level ontology to facilitate interoperability of provenance information in scientific applications. The description logic (DL) expressivity of *provenir* ontology is \mathcal{ALCH} , that is, it models role hierarchies (\mathcal{H}) (without transitive roles and inverse roles). Even though the complexity results for concept satisfiability for numerous variants of DL such as \mathcal{ALC} with transitively closed roles (\mathcal{ALC}_{R^+} also called \mathcal{S}), inverse roles \mathcal{SI} , and role hierarchy \mathcal{SHI} have been well-established, similar results for \mathcal{ALCH} has been surprisingly missing from the literature. Here, we show that the complexity of the concept satisfiability problem for the \mathcal{ALCH} variant of DL is PSPACE complete. This result contributes towards a complete set of complexity results for DL variants and establishes a lower bound on complexity for domain-specific provenance ontologies that extend *provenir* ontology.

Keywords: Provenir ontology, Provenance Management, Concept Satisfiability for DL- \mathcal{ALCH} , Tableau Algorithm

1 Introduction

Provenance information, derived from the French word “provenir”, describes the history or lineage of data and is critical metadata to validate the quality as well as associate trust values to scientific data. The *provenir* ontology [1] is an upper level ontology that can be extended to model domain-specific provenance and thus facilitate interoperability of provenance information. *Provenir* ontology is modeled using OWL-DL, a flavor of the W3C Web Ontology Language (OWL). The DL expressivity of *provenir* ontology is \mathcal{ALCH} , that is, it models role hierarchy (\mathcal{H}), in addition to concept expressions that are closed under negation, union, intersection, limited universal and existential quantification involving atomic roles (\mathcal{ALC}). Computational characteristics of DL variants, extending

* These authors contributed equally to this work

\mathcal{ALC} with (i) role transitivity (\mathcal{ALC}_{R^+} also called S), (ii) with role transitivity and inverse roles (SI), and (iii) with role hierarchy added to SI (SHI) [2] [3] [4] [5] et cetera have been well studied. But, the computational complexity of concept satisfiability for DL- \mathcal{ALCH} alone (that is, without role transitivity, inverse roles, role chaining, and functional roles), has not been explicitly reported. This result is important to define a lower bound for complexity of provenance ontologies that extend *provenir* ontology to model domain-specific provenance in scientific applications.

In this paper, we define the syntax and the semantics of DL- \mathcal{ALCH} , and then provide a sound and complete tableau algorithm for determining concept satisfiability. We then analyze the computational complexity of the tableau algorithm. We know from [2] that the satisfiability problems for DL- \mathcal{ALC} are PSPACE complete. We now prove that the complexity of the concept satisfiability problem for the DL- \mathcal{ALCH} is also PSPACE complete by adapting relevant proofs for \mathcal{ALC} variant of DL given in [2] to accommodate the consequences of role hierarchy without overstepping the polynomial bound on space.

2 DL- \mathcal{ALCH} : Syntax and Semantics

We describe DL- \mathcal{ALCH} by extending the “basic” DL- \mathcal{ALC} [2] with role hierarchy (\mathcal{H}). It uses concept names and propositional constructors such as intersection \cap , union \cup , and negation \neg , and standard quantifiers such as existential \exists and universal \forall with role names, to build complex concepts. Specifically, quantifiers are used with atomic roles to link concepts. The subset \subseteq constructor is used with atomic roles to create a role hierarchy. Note that DL- \mathcal{ALCH} does not permit role expressions.

Definition 1:

Let UC be a set of concept names, UR be a set of role names, $A \in UC$ and $R \in UR$. The concept expressions in DL- \mathcal{ALCH} are built inductively as follows:

$$C ::= A \mid \neg C \mid C_1 \cap C_2 \mid C_1 \cup C_2 \mid \exists R. C \mid \forall R. C$$

The (acyclic) role hierarchy is a collection of role inclusions such as:

$$R_1 \subseteq R_2.$$

For example, a concept expression with role hierarchy is:

$$\exists R_1. C_1 \cap \forall R_2. (C_2 \cup \neg C_3) \text{ with } \{R_1 \subseteq R_2, R_2 \subseteq R_3\}$$

The *size* of the complex concept expression, $|C|$, and the size of the role hierarchy, $|RH|$, are the number of symbols necessary to write down C and RH over the alphabet $UC \cup UR \cup \{\cap, \cup, \neg, \exists, \forall, (,)\}$.

The semantics of the language is defined by induction over its syntactic structure.

Definition 2:

The semantics of DL- \mathcal{ALCH} concepts is defined relative to an interpretation $I = (\Delta^I, \cdot^I)$, where Δ^I is a non-empty set, called domain of I , and \cdot^I is a valuation that maps every concept name to a subset of Δ^I and every role name to a subset of $\Delta^I \times \Delta^I$. The interpretation can be lifted to concept expressions and satisfied by role inclusion (role hierarchy) as follows:

1. $(\neg C)^I = \Delta^I - C^I$
2. $(C_1 \cup C_2)^I = C_1^I \cup C_2^I$
3. $(C_1 \cap C_2)^I = C_1^I \cap C_2^I$
4. $(\forall R. C)^I = \{x \in \Delta^I \mid \forall y \in \Delta^I : \text{if } (x, y) \in R^I \text{ then } y \in C^I\}$
5. $(\exists R. C)^I = \{x \in \Delta^I \mid \exists y \in \Delta^I : (x, y) \in R^I \text{ and } y \in C^I\}$
6. $(R_1 \sqsubseteq R_2)^I = R_1^I \sqsubseteq R_2^I$

3 Tableau Algorithm for DL- \mathcal{ALCH} concept satisfiability

We develop an algorithm that constructs a model for a concept expression C with role hierarchy RH if one exists. This is done by defining an interpretation I such that for the given satisfiable concept C and role hierarchy RH , $C^I \neq \emptyset$ and $\forall ri \in RH: (ri)^I$ holds, where ri stands for role inclusion. Role hierarchy can be viewed as either a conjunction of role inclusions or a set of role inclusions. In what follows, we will minimize the explicit mention of role hierarchy RH and assume its presence in the background without jeopardizing clarity.

a) **Constraint System**

The tableau algorithm constructs a pre-model [2] or a pre-tableau [4] for concept C using a data structure called the *constraint system*, by initializing it with the ABox assertion, $x_0:C$, and then augmenting it using the consequences of completion rules (discussed below) until there are no more rules to apply or a *clash* is detected.

Definition 3:

Let UI be the set of individuals. A constraint system CS is said to contain a clash iff for a concept $C \in UC$ and an instance $x \in UI$, $\{x : C, x : \neg C\} \subseteq CS$ holds. Otherwise, CS is said to be clash-free.

Note that rule hierarchy does not contribute to a clash because both C and RH contain only atomic role names.

b) Negation Normal Form

For convenience, we assume that the concept expression is in negation normal form (NNF), that is, negation is applied to only concept names. Every DL- \mathcal{ALCH} concept expression can be transformed into an equivalent concept expression in NNF by applying the following rules repeatedly:

1. $\neg(C_1 \cup C_2) \equiv \neg C_1 \cap \neg C_2$
2. $\neg(C_1 \cap C_2) \equiv \neg C_1 \cup \neg C_2$
3. $\neg \forall R. C \equiv \exists R. \neg C$
4. $\neg \exists R. C \equiv \forall R. \neg C$
5. $\neg \neg C \equiv C$

For example, the negation of $\exists R_1. C_1 \cap \forall R_2. (C_2 \cap \neg C_3)$ with $\{R_1 \subseteq R_2, R_2 \subseteq R_3\}$ is $(\forall R_1. \neg C_1) \cup (\exists R_2. (\neg C_2 \cap C_3))$ with $\{R_1 \subseteq R_2, R_2 \subseteq R_3\}$

Note that rule hierarchy is not affected by negation because it contains only atomic role names. The size of a concept expression in NNF is linearly related to (in fact, at most twice) the size of the original expression.

c) Completion Rules for DL- \mathcal{ALCH}

We adapt the approach in [2] to prove the satisfiability of a DL- \mathcal{ALCH} concept expression C in NNF. The constraint set \mathcal{A} is initialized with a single assertion $\mathcal{A} = \{x_0 : C\}$. This set is then expanded using the completion rules defined below, until either a clash is found, or the rules can no longer be applied. Note that the algorithm is non-deterministic in two ways: (i) due to the OR-rule (“*don’t know non-determinism*”) and (ii) due to the freedom in sequencing rule applications (“*don’t care non-determinism*”).

We prove that the concept expression C is satisfiable *if and only if* there exists a stable constraint system \mathcal{A} that is clash-free (by extending it to construct a model for C with RH). (Note that Kleene star over the role inclusion operation is to propagate pairs through the role hierarchy.)

\rightarrow_{\cap} if 1. $x: (C_1 \cap C_2) \in \mathcal{A}$ and
 2. $\{x: C_1\} \not\subseteq \mathcal{A}$
 then $\mathcal{A} \rightarrow_{\cap} \mathcal{A} \cup \{x: C_1\}$

AND-Rule1

\rightarrow_{\cap} if 1. $x: (C_1 \cap C_2) \in \mathcal{A}$ and
 2. $\{x: C_2\} \not\subseteq \mathcal{A}$
 then $\mathcal{A} \rightarrow_{\cap} \mathcal{A} \cup \{x: C_2\}$

AND-Rule2

\rightarrow_{\cup} if 1. $x: (C_1 \cup C_2) \in \mathcal{A}$ and
 2. $\{x: C_1, x: C_2\} \cap \mathcal{A} = \emptyset$
 then $\mathcal{A} \rightarrow_{\cup} \mathcal{A} \cup \{x: \mathcal{A}\}$ for some \mathcal{A}
 $\in \{C_1, C_2\}$

OR-Rule

\rightarrow_{\exists} if 1. $x: \exists R. B \in \mathcal{A}$ and
 2. there is no y such that $(x, y): R$
 $\in \mathcal{A}$ and $y: B \in \mathcal{A}$
 then $\mathcal{A} \rightarrow_{\exists} \mathcal{A} \cup \{(x, y): R, y: B\}$
 $\cup \{(x, y): R_j \mid R \sqsubseteq_* R_j$
 $\in RH\}$ for a **new** y

EXISTS-Rule

\rightarrow_{\forall} if 1. $x: \forall R. B \in \mathcal{A}$ and
 2. there is a y such that $(x, y): R \in \mathcal{A}$ and $y: B$
 $\notin \mathcal{A}$

FORALL-Rule

then $\mathcal{A} \rightarrow_{\forall} \mathcal{A} \cup \{y: B\}$

(d) Proof of Correctness of the Algorithm

In this section, we show (i) termination, (ii) soundness, and (iii) completeness of the DL- \mathcal{ALCH} tableau algorithm.

i) **Termination**: To show that the tableau algorithm terminates, we use an auxiliary graph structure G that can be induced from an ABox \mathcal{A} , the set of “ground” assertions involving concept expressions and role names [2].

Definition 4

Given an ABox \mathcal{A} , the graph $G_{\mathcal{A}}$ is a directed, node and edge labeled graph, with nodes (N), edges (E), and a labeling function (L), such that,

$$N = \{ x \in UI \mid x \text{ occurs in } \mathcal{A} \},$$

$$E = \{ (x, y) \mid (x, y): R \in \mathcal{A} \},$$

$$L(x) = \{ C \mid x: C \in \mathcal{A} \},$$

$$L(x, y) = \{ R \mid (x, y): R \in \mathcal{A} \}.$$

Conceptually, graph $G_{\mathcal{A}}$ can be viewed as providing an “object-oriented perspective” on the patently “relational” ABox \mathcal{A} .

Lemma 1

Given a concept expression C of DL- \mathcal{ALCH} in NNF and an ABox \mathcal{A} generated by the application of completion rules from the initial seed $x_0: C$, the following facts hold of the graph $G_{\mathcal{A}}$:

- a) *The size of a node label set $L(x)$ is bounded by $|C|$, for every node x in $G_{\mathcal{A}}$.*
- b) *The total length of a path in $G_{\mathcal{A}}$ is bounded by $|C|$.*
- c) *The out-degree of a node in $G_{\mathcal{A}}$ is bounded by $|C|$.*
- d) *The size of the edge label set $L(x, y)$ is bounded by $|RH|$, for every edge (x, y) in $G_{\mathcal{A}}$.*

Proof:

The proof can be given by induction on the number of rule applications similarly to DL- \mathcal{ALC} [2].

For (a), observe that AND-rules and OR-rule add sub-concepts to the node label, and EXISTS-rule and FORALL-rule can add sub-concepts to a different node label by virtue

of direct successors spawned by the first individual. In both cases, the number of elements in the node label is bounded by the size of the original concept expression C . (The effect of RH is accounted for by occurrences of role names in C .) For example, consider the consequences A of

$$\begin{aligned}
 & x: \exists R.B \cap \forall R_1.B_1 \cap \forall R_2.B_2 \\
 & \text{with } R \subseteq R_1, R_1 \subseteq R_2 \\
 A \supseteq & \{ x: \exists R.B, \quad x: \forall R_1.B_1, \quad x: \forall R_2.B_2, \\
 & (x,y): R, \quad y: B, \quad (x,y): R_1, \quad (x,y): R_2, \\
 & \quad \quad \quad y: B_1, \quad y: B_2 \}
 \end{aligned}$$

For (b), observe that when a successor to an individual node is created, the corresponding concept label has one less quantifier than the related member of the label on the predecessor. Thus, the path from root individual x_0 to a leaf can be strictly ordered on the basis of maximum number of quantifiers in a member of the label. For example, the $\exists R.B \in L(x)$ and the corresponding member obtained for the successor node y is B , that is, $B \in L(y)$ which has one less quantifier.

For (c), observe that each successor is spawned by an existential quantifier and the number of quantifiers in C is bounded.

For (d), observe that the size of the edge label set is bounded by the number of role names.

Corollary 1

A sequence of rule applications of the tableau algorithm for DL- \mathcal{ALCH} will terminate after a finite number of steps.

Proof: Each rule application either adds a member to a node label set and/or an edge label set, or creates a new edge by allocating a successor node. The size of the node label sets and edge label sets, and the number of nodes and edges, are all bounded by $|C+RH|$. No nodes, edges, or labels are ever deleted. So these sets must stabilize and the algorithm terminates.

ii) **Soundness:** We demonstrate soundness by showing that if the ABox $\{x_0:C\}$ with RH is satisfiable then there exists a stable constraint system \mathcal{A} obtained by applying the completion rules of the DL- \mathcal{ALCH} tableau-algorithm, that is satisfiable, and hence, clash-free.

Proof: The proof is by induction on the number of iterations to obtain the stable set \mathcal{A} by verifying that each application of a completion rule preserves satisfiability. It is easy to check for each completion rule that the consequents of (that is, the assertions added by) each rule (AND-rules, OR-rule, EXISTS-rule and FORALL-rule) must hold if the antecedent holds, given the semantics of the various operations. Specifically, note the

“*don't know non-determinism*” due to the OR-rule that makes the algorithm non-deterministic, and the fact that EXISTS-rule introduces a new individual that does not interfere with existing assertions in the constraint system (because there are no qualifying number restrictions). Note also that, in the presence of role hierarchy, the EXISTS-rule requires propagating the pairs in relation R to all relations that include R. (For convenience of proving soundness, we did not add an additional EXISTS-rule for situation where $\exists x: R.B$ and $(x, y): R$ already hold, and hence requiring only $y \in R$ to be added, as opposed to creating a new individual z. As such, EXISTS-rule may introduce more individuals than are absolutely necessary.)

For example, the consequences of $x: \exists R_1.C_1 \cap \exists R_2.C_2$ with $R_1 \sqsubseteq R_2$ can contain $\{(x, y): R_1, y: C_1, (x, y): R_2, (x, z): R_2, z: C_2\}$, for the newly introduced y and z. Note also that one can construct a model with fewer individuals (without introducing z) which contains $\{(x, y): R_1, y: C_1, (x, y): R_2, y: C_2\}$.

iii) **Completeness:** We demonstrate completeness by constructing a *model* for ABox $\{x_0:C\}$ with RH by potentially extending the stable clash-free constraint system \mathcal{A} obtained by the DL- \mathcal{ALCH} tableau-algorithm.

Proof: Let N be the total number of rule applications to obtain the stable set \mathcal{A} , and let \mathcal{B}_j be the subset of assertions \mathcal{A} created after j rule applications. We show by induction on the quantity $i = (N - j)$ (resp. j) that ultimately \mathcal{B} (resp. RH) are satisfied by the canonical interpretation defined below.

Canonical Interpretation

*For an Abox \mathcal{A} , the canonical interpretation $I_{\mathcal{A}}$
 $= (\Delta^{I_{\mathcal{A}}}, \cdot^{I_{\mathcal{A}}})$ is defined by*

$$\Delta^{I_{\mathcal{A}}} = \{x \in UI \mid x \text{ occurs in } \mathcal{A}\},$$

$$A^{I_{\mathcal{A}}} = \{x \mid x: A \in \mathcal{A}\} \text{ for every } A \in UC,$$

$$R^{I_{\mathcal{A}}} = \{(x, y) \mid (x, y): R \in \mathcal{A}\} \text{ for every } R \in UR,$$

$$x^{I_{\mathcal{A}}} = x \text{ for every individual } x \text{ that occurs in } \mathcal{A}$$

Since \mathcal{A} is clash-free, $x: \neg A \in \mathcal{A}$ implies $x: A \notin \mathcal{A}$. Thus, $x \notin \mathbf{A}^I \mathcal{A}$

Basis: $i = 0$. $\mathcal{B}N$ is satisfied by the canonical interpretation, as $\mathcal{B}N$ contains consequents obtained in the last step, which must be of the form $x:C$ or $(x, y):R$. Otherwise, a completion rule will be applicable to the consequents and the algorithm cannot terminate.

Induction Hypothesis: Assuming that $\mathcal{B}j+1$ is satisfied by the canonical interpretation; show that $\mathcal{B}j$ is also satisfied by the canonical interpretation.

Induction Step: Let $j = N - i$. Assuming that \mathcal{B} is obtained from $\mathcal{B}j$ by the application of a completion rule. If the consequent of that completion rule is satisfied by the canonical interpretation, then one can verify that the antecedent of that rule is also satisfied by the same interpretation.

- If AND-rule was applied in j^{th} step such that $x:C_1 \cap C_2 \in \mathcal{B}j$ and $\{x:C_1, x:C_2\} \subseteq \mathcal{B}j+1$, then, by induction this implies $x \in C_1^I_{\mathcal{A}}$ and $x \in C_2^I_{\mathcal{A}}$ and hence $x \in (C_1 \cap C_2)^I_{\mathcal{A}}$
- If one of the OR-rule was applied in j^{th} step such that $x:C_1 \cup C_2 \in \mathcal{B}j$, and, either $x:C_1 \in \mathcal{B}j+1$ or $x:C_2 \in \mathcal{B}j+1$, then, by induction $x \in C_1^I_{\mathcal{A}}$ or $x \in C_2^I_{\mathcal{A}}$ and hence $x \in (C_1 \cup C_2)^I_{\mathcal{A}}$
- If EXISTS-rule was applied in the j^{th} step such that $x: \exists R.A \in \mathcal{B}j$, and $\{(x, y): R, y:A\} \cup \{(x, y): R_j \mid R \sqsubseteq_* R_j \in RH\} \subseteq \mathcal{B}j+1$ for some y . Then, by construction of $I_{\mathcal{A}}$, $(x, y) \in R^I_{\mathcal{A}}$ holds, and for every R_c that includes R according to RH , $(x, y) \in R_c^I_{\mathcal{A}}$ holds. By induction we have $y \in A^I_{\mathcal{A}}$. Together this implies $x \in (\exists R.D)^I_{\mathcal{A}}$
- If FORALL-rule was applied in j^{th} step such that $x:\forall R.A \in \mathcal{B}j$, and, the fact that $(x, y):R \in \mathcal{B}j$, it also follows that $(x, y) \in R^I_{\mathcal{A}}$, $(x, y): R \in \mathcal{B}j+1$ must hold due to the construction of $I_{\mathcal{A}}$. Then, due to completeness, $y:A \in \mathcal{B}j+1$ must hold and induction yields $y \in A^I_{\mathcal{A}}$. Since this holds for any such y , $x \in (\forall R.D)^I_{\mathcal{A}}$

In order to see that the canonical interpretation satisfies RH, note that every application of EXISTS-rule to a concept expression such as $x: \exists R.A$, introduces a new element (x,y) into the relation R that is immediately propagated to all relations that include it $\{(x,y): R_j \mid R \subseteq_*, R_j \in RH\}$, ensuring that RH is satisfied. This informal argument can be easily formalized by inducting on the number of steps required to reach the stable constraint set. (Observe that the proof of satisfiability of concept expression goes “backwards”, while the proof of satisfiability of rule hierarchy goes “forwards”.)

3.1 Computational Complexity

We adapt the approach of [2] for demonstrating the PSPACE-completeness of the concept satisfiability problem of DL- \mathcal{ALC} to prove that the result carries over to DL- \mathcal{ALCH} . A naïve implementation of the concept satisfiability algorithm described above requires construction of a model that is exponential in the size of the input concept expression, in the general case [6]. However, to obtain optimal worst-case complexity, the trace technique can be used for DL- \mathcal{ALCH} (similarly to DL- \mathcal{ALC} , and unlike DL with role inverses), because it is sufficient to consider a single path in G_A at any time, leading to efficient reuse of space.

We first modify the non-deterministic PSPACE decision process of \mathcal{ALC} to work for \mathcal{ALCH} :

Figure 1: A non-deterministic PSPACE decision procedure for \mathcal{ALCH}

$ALCH_SAT(C) := sat(x_0, \{x_0: C\})$

$sat(x, \mathcal{A})$:

while (*the \rightarrow_{\cap} or the \rightarrow_{\cup} rule can be applied*)

and (\mathcal{A} is clash_free) **do**

apply the \rightarrow_{\cap} or the \rightarrow_{\cup} rule to \mathcal{A}

od

if \mathcal{A} contains a clash **then return** not satisfiable.

$$E := \{x: \exists R.D \mid x: \exists R.D \in \mathcal{A}\}$$

while $E \neq \emptyset$ **do**

pick an arbitrary $x: \exists R.D \in E$

$$\mathcal{A}_{new} := \{(x, y): R, y: D\} \cup \{(x, y): R_j \mid R \sqsubseteq_* R_j \in RH\}$$

where y *is a fresh individual*

while (the \rightarrow_{\forall} can be applied to $\mathcal{A} \cup \mathcal{A}_{new}$) **do**

apply the \rightarrow_{\forall} *rule and add new constraints to* \mathcal{A}_{new}

od

if $\mathcal{A} \cup \mathcal{A}_{new}$ *contains a clash* **then return** not satisfiable

if $\text{sat}(y, \mathcal{A} \cup \mathcal{A}_{new}) = \text{not satisfiable}$ **then return** not satisfiable

$$E := E - \{x: \exists R.D \mid (x, y): R \in \mathcal{A}_{new} \wedge y: D \in \mathcal{A}_{new}\}$$

(*i.e., remove the existential assertion just explored

$x: \exists R.D$ from further consideration*)

discard \mathcal{A}_{new} *from memory*

od

return "satisfiable".

Lemma 2

The concept satisfiability algorithm for DL- \mathcal{ALCH} requires PSPACE.

Proof: Let C be the DL- \mathcal{ALCH} concept expression with role hierarchy RH to be tested for satisfiability. C is assumed to be in NNF because transformation into NNF is linear in

time and space. Figure 1 sketches the implementation of the \mathcal{ALCH} algorithm that uses the trace technique [7] to run in polynomial space.

The algorithm generates the constraint system in a depth-first manner. Before generating any successors for an individual x , the AND-rules and the OR-rule are applied exhaustively, effectively augmenting \mathcal{A} with labels associated with individual node x in $G_{\mathcal{A}}$. Then, the successors are considered for every existential restriction in \mathcal{A} one after another re-using space. If a clash involving an individual x is not present in \mathcal{A} by the time generation of successors of x is initiated, it will never occur. This is easy to see by considering the nature of the node labels in $G_{\mathcal{A}}$ and the fact that role hierarchy RH contributes only positive edge labels to $G_{\mathcal{A}}$. Hence, it is safe to delete parts of the constraint system for a successor y as soon as the existence of complete and clash-free “sub” constraint system has been determined. Similar to [2] it is important to ensure that the same existential restriction $x: \exists R.D$ is not considered more than once; else it will lead to non-termination. The algorithm records the constraints that have not been explored so far in the set \mathbf{E} .

The algorithm resets \mathcal{A}_{new} for every explored successor, effectively storing all the members of node label sets (concept sub-expressions) and edge label sets (role assertions) on a single path explicitly in \mathcal{A} . A simple estimate of the maximum size and the space requirements of the set \mathcal{A} in terms of the size of the original concept expression C and role hierarchy RH can be obtained as follows:

$$\text{Max. size of each concept in node label set} = O(|C|)$$

$$\text{Max. size of a node label set} = O(|C|)$$

$$\text{Max. length of a path in } G_{\mathcal{A}} = O(|C|)$$

$$\text{Total space for node labels on a path} = O(|C|^3)$$

$$\text{Max. size of edge label set} = O(|RH|)$$

$$\text{Total space for edge labels on a path} = O(|RH| * |C|)$$

$$\text{Total space for the set } \mathcal{A} = O(|C|^3 + |RH| * |C|)$$

Thus, the space complexity for the concept satisfiability for DL- \mathcal{ALCH} is polynomial.

Theorem 1

The complexity of concept satisfiability for DL- \mathcal{ALCH} is PSPACE-complete.

Proof:

Given that the concept satisfiability problem for DL- \mathcal{ALC} (and hence DL- \mathcal{ALCH}) is PSPACE-hard [7], and *Lemma 2* demonstrates that the concept satisfiability problem for DL- \mathcal{ALCH} is in PSPACE, the PSPACE-completeness of concept satisfiability problem for DL- \mathcal{ALCH} follows.

4 Conclusions

The DL expressivity of *provenir*, an upper-level ontology for provenance information, is \mathcal{ALCH} . Even though the computational complexity of concept satisfiability for many DL variants including *S*, *SI*, and *SHI* has been widely known, similar result for DL- \mathcal{ALCH} has been conspicuously absent from the literature. We have now proved that the complexity of concept satisfiability for DL- \mathcal{ALCH} is PSPACE-complete. This result fills an obvious gap in the extensive computational complexity results for DL variants and establishes a lower bound for provenance ontologies that extend *provenir* ontology to model domain-specific provenance in scientific applications.

Acknowledgement

This work is funded by NIH RO1 Grant# 1R01HL087795-01A1.

References

- [1] Sahoo SS, Barga, R.S., Goldstein, J., Sheth, A.P., Thirunarayan, K. "Where did you come from...Where did you go?" An Algebra and RDF Query Engine for Provenance Kno.e.sis Center, Wright State University; 2009.
- [2] Tobies S. Complexity Results and Practical Algorithms for Logics in Knowledge Representation. : Ph.D. Thesis, RWTH Aachen, Germany; 2001.
- [3] Baader F, Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F, editor. The Description Logic Handbook: Theory, Implementation, and Applications: Cambridge University Press; 2003.

- [4] Horrocks I, Sattler, U., Tobies, S. Practical reasoning for expressive description logics. In: Harald Ganzinger DM, and Andrei Voronkov, editor. 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99); 1999: Springer-Verlag; 1999. p. 161-180.
- [5] <http://www.cs.man.ac.uk/~ezolin/dl/>.
- [6] Halpern JY, Moses, Y. A guide to completeness and complexity for model logics of knowledge and belief. *Artificial Intelligence* 1992;54(3):319-379.
- [7] Schmidt-SchauB M, Smolka, G. Attributive concept descriptions with complements. *Artificial Intelligence* 1991;48:1-26.