ISSN 2007-9737

(Hyper)sequent Calculi for the ALC(S4) Description Logics

Juan Pablo Muñoz[1], Everardo Bárcenas[2], Iván Martínez[1],
José Ramón Enrique Arrazola[1]

[1] Benemérita Universidad Autónoma de Puebla, FCFM,
Mexico

[2] CONACYT - Universidad Veracruzana,
Mexico

juanpablo.munoz@alumno.buap.mx, iebarcenaspa@conacyt.mx,
{imartinez, arrazola}@fcfm.buap.mx

Abstract. Description logics (DL) form a well-known family of knowledge representation languages. One of its main applications is on the Semantic Web as a reasoning framework in the form of the Ontology Web Language (OWL). In this paper, we propose a cut-free tree hypersequent calculus for terminological reasoning in the Description Logic ALC. We show the calculus is sound and complete. Also, an implementation is provided together with a complexity analysis. In addition, we also describe a cut-free sequent calculus for the description logic ALC with reflexive and transitive roles. Soundness and completeness are proven, and a complexity analysis and an implementation are also provided.

Keywords. Description logics, (Hyper)sequents, proof theory, automated reasoning.

1 Introduction

Description logic languages (DL) are nowadays a well-established formalism for knowledge representation [1]. Closing the gap between theory and practice is one of the most appreciated features in DL. This is well exemplified by the success of the Web Ontology Language (OWL) and related reasoning technologies in the Semantic Web [11].

On the theoretical side, from the seminal work of Schild [17], a close relationship between modal and description logics is know, namely, that basic propositionally closed concept language ALC is a notational variant of the multi-modal logic K_m. Other relationships between more expressive logics such as propositional dynamic logic and ALC_{reg} were later reported in [6]. These observations have been very useful in the translation of results, mostly regarding computational complexity, from the modal family to the description one [12]. However, mainly due to the semantic tradition of modal logic, there is still an overall perception of unsatisfactory proof theory on description logics [14].

The formal deductive systems of sequents for first order classical and intuitionistic logics were first developed by Gentzen in his search for arithmetic consistency [7]. Since then, sequent calculi have been an important tool in the syntactic analysis of proofs and in the automation of reasoning tasks.

More recent generalization of sequents systems such as (tree) hypersequents have lead to important proof theoretic results on large classes of logics, not including description ones [9, 14, 15]. In this paper, we describe a sequent system for ALC with reflexive and transitive roles, that is, the notational variant of the modal logic S4. A complexity analysis of this system is also provided together with an implementation.

Also, a tree hypersequent system for ALC with general roles, i.e., the notational variant of the modal logic K, is also described together with its complexity analysis and implementation.

1.1 Motivations and Related Work

The proof theory of modal logic is not as well understood as its classical counterpart [15, 14]. Negri outlines very well in [14] the development and difficulties in the syntactic analysis of proofs for modal logics. In Poggiolesi's thesis [15], there is a detailed description from the sequent systems perspective. In particular, a tree hypersequent system for the modal logic K is proposed, which is a notational variant from the system proposed in the current work for ALC. We, in addition, provide a complexity analysis of the system and an implementation.

Regarding the proof theory of description logic, we find the works of Rademaker [16], and more recently, Su and Sui [19]. In this last work, a sequent system for ALC is prpoposed. However, cut elimination does not hold, and the system is undecidable. In [16], a cut-free sequent system is also proposed for ALC, and counter-models may be extracted from unsuccessful proofs. In the current work, we propose a cut-free deduction system based on tree of hypersequents, which in addition is contraction-free. This allows an implementation of the system, which is also provided.

We also propose a cut and contraction free sequent system for ALC with reflexive and transitive roles, which we name ALCS4 due to its obvious correspondence with the multi-modal logic S4. We also provide an implementation and a complexity analysis of this system.

1.2 Contributions and Outline

In Section 2, we describe the basic propositionally closed concept language ALC. Axiomatizations for ALC with general, and reflexive and transitive roles are also provided. A cut and contraction free sequent system for ALC with reflexive and transitive roles is described in Section 3. Also, the system is proven correct and to be in EXPTIME. An implementation of the system is also described. In Section 4, we describe a tree hypersequent system for ALC. The system is also cut and contraction free, which allows to implement it. The implementation is also given in this Section. Correctness and complexity proofs are described. Finally, in Section 5, we give an outline of the current work together with a discussion of further research perspectives.

2 Description Logics

In this Section, we describe two concept languages, the basic propositionally closed concept language ALC introduced by Schmidt-Schaußand Smolka [18], and a restriction of ALC with reflexive and transitive roles, which we name ALCS4 due to its correspondence to multi-modal logic S4. We define the syntax and semantics of the logics, and we also describe the corresponding axiomatizations, which are used in the completeness proofs of the derivation system proposed in the current paper.

2.1 Syntax and Semantics

We assume a fixed alphabet language composed by sets of concept and role names.

Definition 1 (Syntax). Concept descriptions in the basic propositionally closed concept language (ALC) are given by the following grammar:

$$C := A \mid \neg C \mid C \sqcap C \mid \forall r.C,$$

where A is a concept name and r is a role name.

Concepts names are interpreted on relational structures as node subsets: names A are used as node labels; negation $\neg C$ is interpreted as set complement; semantics for conjunction $C \sqcap D$ corresponds to set intersection; and concepts $\forall r.C$ denote nodes where all their accessible nodes through the role r satisfy C.

We also consider the following notation: $C \sqcup D := \neg(\neg C \sqcap \neg D)$, $\bot := \neg A \sqcap A$, $\top := \neg\bot$, $C \sqsubseteq D := \neg C \sqcup D$, $C \equiv D := (C \sqsubseteq D) \sqcap (D \sqsubseteq C)$, and $\exists r.C := \neg\forall r.\neg C$. Note then that disjunctions $C \sqcup D$ are interpreted as set unions, $\bot$ and $\top$ denote contradictions and tautologies, respectively, concepts $C \sqsubseteq D$ are interpreted as implications, $C \equiv D$ as double implication, and the semantics of concepts $\exists r.C$ corresponds to nodes where there is at least one accessible node through r satisfying C.

Consider for instance the following concept description [5]:

$$\text{Man} \sqcap \exists\text{married.Doctor} \sqcap \forall\text{child.Happy}.$$

This concept description can be interpreted as men married with a doctor and with happy children only. In Figure 1, a model (as a relational structure) of this concept is depicted.

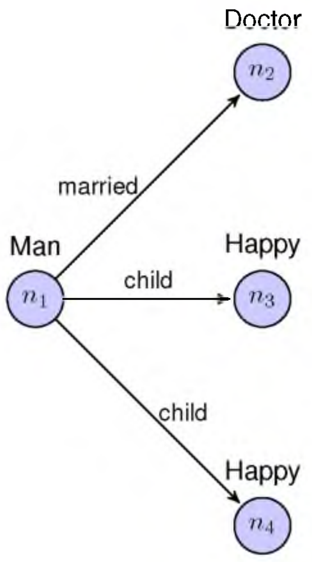

Fig. 1. A relational model for Man $\sqcap$ $\exists$married.Doctor $\sqcap$ $\forall$child.Happy holding at n_1

Before giving a formal description of ALC semantics we first introduce the notion of interpretation, which is a pair $I = (\Delta^I, \cdot^I)$, where Δ^I is a non-empty set called the domain of I, and $\cdot^I$ is the interpretation function, which assings to each concept name A a domain subset $A^I \subseteq \Delta^I$ and to each role name r a binary relation on the domain $r^I \subseteq \Delta^I \times \Delta^I$.

Definition 2 (Semantics)**.** Given an interpretation $I = (\Delta^I, \cdot^I)$, a semantics of concept descriptions is defined as follows.

$$\begin{aligned}(\neg C)^I =& \Delta^I \setminus C^I,\\ (C \sqcap D)^I =& C^I \cap D^I,\\ (\forall r.C)^I =& \left\{d \in \Delta^I \mid \forall e.(d,e) \in r^I \to e \in C^I\right\}.\end{aligned}$$

An interpretation I is a model of a concept description C when $C^I \neq \emptyset$, and it is written $I \models C$, we also say C is satisfiable by I. If any interpretation is a model C, then we say C is valid, and we write $\models C$.

2.2 ALC Axiomatization

It is well-known that ALC is a notational variant of multi-modal logic K [17], hence, an axiomatization for K is also an axiomatization for ALC, which is composed by the axioms of classical logic together with the normality scheme and the rules of Modus Ponens and Necessitation.

We now give an axiomatization of ALC as given in [16].

Definition 3 (ALC axiomatization)**.** Axioms are given as follows:

A1 $\neg(C \sqcap (D \sqcap \neg C))$,

A2 $\neg((\neg(C \sqcap (D \sqcap (\neg E)))) \sqcap ((\neg(C \sqcap (\neg D))) \sqcap (C \sqcap (\neg E))))$,

A3 $\neg((\neg((\neg C) \sqcap D)) \sqcap ((\neg((\neg C) \sqcap (\neg C))) \sqcap (\neg C)))$,

A4 $\neg((\forall r.(\neg(C \sqcap (\neg D)))) \sqcap ((\forall r.C) \sqcap (\neg(\forall r.D))))$.

Inference rules are Modus Ponens (MP) and Necessitation (Nec).

$$\frac{\neg(C \sqcap \neg D) \qquad C}{D}\, MP, \qquad \frac{C}{\forall r.C}\, Nec.$$

The axioms A1-A3 can be understood as a notational variant of the corresponding axioms in classical propositional calculus, while Axioma A4 corresponds to normality in the multi-modal logic K.

If a concept description C is derivable with the ALC axiomatization, then we write $\vdash C$.

The correctness of the ALC axiomatization trivially follows.

Theorem 1 ([5, 16])**.** *ALC axiomatization is sound and complete, that is, for any ALC concept description C, we have that*

$$\vdash C \textit{ if and only if } \models C.$$

ISSN 2007-9737

2.3 ALCS4

Given a binary relation R over a set S, we say R is reflexive if and only if for all $s \in S$, we have that $(s, s) \in R$. In the same context, we say that R is transitive, if and only if, for all $s_1, s_2, s_3 \in S$, we have that if $(s_1, s_2) \in R$ and $(s_2, s_3) \in R$, then $(s_1, s_3) \in R$.

In the setting of ALC concept descriptions, we say an interpretation $I = (\Delta^I, \cdot^I)$ is reflexive (transitive), if and only if, for any role r, we have that r^I is reflexive (transitive) over Δ^I.

Definition 4 (ALCS4). ALCS4 is the logic of concept descriptions described by the ALC syntax (Definition 1), but whose semantics (Definition 2) considers reflexive and transitive interpretations only.

As expected, the axiom system for ALCS4 corresponds to the axiom system of the multi-modal logic S4.

A usual way to study a formal logic consists in defining a set of axioms and inference rules and then, if possible, obtaining an adequate semantic system for it. However, to study DL, it is more natural to proceed in the opposite direction, i.e., starting from a semantic interpretation we analyze the valid consequences of the corresponding logic and then, if possible, define a set of axioms and rules for it.

Definition 5 (ALCS4 axiomatization). ALCS4 axiomatization is formed by the axioms and inference rules of the ALC axiomatization (Definition 3) together with the following axioms:

A5 $\neg(\neg C \sqcap \forall r.C)$,

A6 $\neg(\neg(\forall r.(\forall r.C)) \sqcap \forall r.C)$.

Abusing notation, when clear from context, we also write $\vdash C$ when an ALCS4 concept description C is derivable from its corresponding axiomatization.

The following results present necessary and sufficient conditions to verify axioms A5 and A6.

Proposition 1. *An interpretation I is reflexive, if and only if, it is a model for the concept description $\forall r.C \sqsubseteq C$ for any concept description C.*

Proof. If $x \in (\forall r.C)^I$, then for all y that $(x, y) \in r^I$, we know that $y \in C^I$.

Now, since I is reflexive, then $(x, x) \in r^I$, therefore, $x \in C^I$.

Let $x \in \triangle^I$ and $(x, x) \notin r^I$. If C is such that $C^I = \triangle^I \setminus \{x\}$, then $x \in (\forall r.C)^I$, hence $x \in C$ which is a contradiction. Therefore $(x, x) \in r^I$. □

Proposition 2. *An interpretation I is transitive, if and only if, I is a model for the concept description $\forall r.C \sqsubseteq \forall r.\forall r.C$ for any concept description C.*

Proof. If $x \in (\forall r.C)^I$, then for all y that $(x, y) \in r^I$, we have that $y \in C$.

Now, since I is transitive, then for all z that $(y, z) \in r^I$, we know that $(x, z) \in r^I$, hence $z \in C^I$ (recall $x \in (\forall r.C)^I$).

Therefore $x \in (\forall r.\forall r.C)$.

Now, let $x, y, z \in \triangle^I$, such that $(x, y) \in r^I$ and $(y, z) \in r^I$, and let us assume that $(x, z) \notin r^I$.

If C is such that $C^I = \triangle^I \setminus \{z\}$, then $x \in (\forall r.C)^I$, hence $x \in (\forall r.(\forall r.C))^I$ and as $(x, y) \in r^I$ y $(y, z) \in r^I$, then $z \in C^I$, which is a contradiction.

We then conclude that $(x, z) \in r^I$. □

Theorem 2. *ALCS4 axiomatization is sound and complete, that is, for any concept description C and any reflexive and transitive interpretation I, we have that*

$$\vdash C \text{ if and only if } I \models C.$$

Proof. We proceed as in the proof of Theorem 1 ([5, 16]), with the additional condition that the rules are reflexive and transitive, which are supported by Propositions 1 and 2. □

3 Sequents for ALCS4

We first introduce in this Section a cut and contraction free sequent system for ALCS4. Then we prove its correctness. A complexity analysis of the sequent system together with an implementation are also provided.

3.1 The Calculus

The notion of sequent is first described. A sequent is a pair (Γ, Λ), written $\Gamma \Rightarrow \Lambda$, where Γ and Λ are sets of concept descriptions, such that both of them cannot be empty. Intuitively, the symbol $\Rightarrow$ denotes that the union of concepts in Γ implies the disjunction of concepts in Λ. We often write $\Gamma, C_1, \ldots, C_k$ instead of $\Gamma \cup \{C_1, \ldots, C_k\}$.

We denote $\bigsqcap \Gamma$ and $\bigsqcup \Delta$ instead of $\bigsqcap_{C \in \Gamma} C$ and $\bigsqcup_{C \in \Delta} C$ respectively.

A sequent rule is a non-empty tuple of sequents $(S_0, S_1, S_2, \ldots, S_n)$, and it is written as follows:

$$\frac{S_1 \quad S_2 \quad \ldots \quad S_n}{S_0}.$$

If a sequent rule contains only one sequent, then it is called an initial sequent. The intuitive meaning of a sequent rule is that the first sequent S_0 is derived from assumption of the other sequents S_i $(i > 0)$.

If the rule is an initial sequent, then that sequent can be interpreted as an axiom, because there is no need of assumptions to derive it. A non-empty set of sequent rules is called a sequent system.

Before defining a sequent system for ALCS4, we define the following notation: $\forall r.\Gamma$ instead of $\{\forall r.C_1, \ldots, \forall r.C_k\}$, where $\Gamma = \{C_1, \ldots, C_k\}$.

Definition 6 (ALCS4 sequent system)**.** We define a sequent system for ALCS4 by the following rules.

$$\frac{}{C, \Gamma \Rightarrow C, \Lambda}\, Ax$$

$$\frac{\Gamma \Rightarrow C, \Lambda}{\neg C, \Gamma \Rightarrow \Lambda}\, \neg A \qquad \frac{C, \Gamma \Rightarrow \Lambda}{\Gamma \Rightarrow \neg C, \Lambda}\, \neg K$$

$$\frac{C, D, \Gamma \Rightarrow \Lambda}{C \sqcap D, \Gamma \Rightarrow \Lambda}\, \sqcap A \qquad \frac{\Gamma \Rightarrow C, \Lambda \quad \Gamma \Rightarrow D, \Lambda}{\Gamma \Rightarrow C \sqcap D, \Lambda}\, \sqcap K$$

$$\frac{C, \Gamma \Rightarrow \Lambda}{\forall r.C, \Gamma \Rightarrow \Lambda}\, \forall A \qquad \frac{\forall r.\Gamma \Rightarrow C}{\forall r.\Gamma \Rightarrow \forall r.C}\, \forall K.$$

Intuitively, a proof is defined as a tree of sequents where each step is obtained by a rule of a given sequent system, and where all the branches are finite and begin with initial sequents.

Consider for instance the following proof of axiom A1 in the sequent system ALCS4.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{C, D \Rightarrow C}\, Ax}{(\neg C), D, C \Rightarrow}\, \neg A}{(D \sqcap (\neg C)), C \Rightarrow}\, \sqcap A}{(C \sqcap (D \sqcap (\neg C))) \Rightarrow}\, \sqcap A}{\Rightarrow (\neg(C \sqcap (D \sqcap (\neg C))))}\, \neg K.$$

A substitution is a function from the set of names concepts to the set of arbitary concepts which has finite support.

Definition 7 (Proof tree)**.** Given a sequent system, a derivation tree is inductively defined as follows:

— any rule of the sequent system, up to substitution, is a derivation tree;

— the following expression is a derivation tree

$$\frac{T_1 \quad \ldots \quad T_n}{S_0}$$

provided that T_i $(i = 1, \ldots, n)$ are derivation trees, and

$$\frac{S_1 \quad \ldots \quad S_n}{S_0}$$

is a rule, up to substitution, such that S_i is the corresponding root (lowest sequent) of T_i.

A derivation tree is a proof tree, or simply a proof, if all its branches are finite and begin with initial sequents. If there is a proof for a sequent $\Rightarrow \Gamma$, then we write $\vdash_G \Gamma$ (G due to Gentzen).

3.2 Correctness

In order to show ALCS4 is correct, we then show soundness and completeness.

We now state the following lemma required to prove soundness.

Lemma 1. *If $\forall r.C \sqsubseteq D$ is valid, then $\forall r.\forall r.C \sqsubseteq \forall r.D$ also does.*

Proof. If $x \in (\forall r.C \sqsubseteq D)^I$ for some interpretation I, then $x \in D^I$ when $x \in (\forall r.C)^I$.

Now, assume the last. There are two cases, when there is a y, such that $(y, x) \in r^I$, and when there is not.

The second case is straightforward since $x \in (\forall r.\forall r.C)^I$.

In the first case, it is clear that $y \in (\forall r.\forall r.C)^I$, now, by assumption we know $x \in D^I$, and thus $y \in (\forall r.D)^I$. □

We are now ready to prove soundness.

Theorem 3 (Soudness). *If there is a proof of sequent $\Gamma \Rightarrow \Lambda$ in ALCS4 sequent system, then formula $\bigsqcap \Gamma \sqsubseteq \bigsqcup \Lambda$ is valid on ALCS4, that is, every reflexive and transitive interpretation is a model of the formula.*

Proof. We proceed by induction on the height of the proof tree of $\Gamma \Rightarrow \Lambda$.

The base case for $C, \Gamma \Rightarrow C, \Lambda$ is trivial.

If it is now assumed that we have a proof

$$\frac{T}{\Gamma' \Rightarrow \Lambda'},$$

we then have to show that if there is also proof

$$\frac{\dfrac{T}{\Gamma' \Rightarrow \Lambda'}}{\Gamma \Rightarrow \Lambda},$$

then $\bigsqcap \Gamma \sqsubseteq \bigsqcup \Lambda$ is valid.

Consider the case when the last step in the proof is the following

$$\frac{\Gamma \Rightarrow C, \Lambda}{\neg C, \Gamma \Rightarrow \Lambda}.$$

Then by induction we have that $\bigsqcap \Gamma \sqsubseteq C \sqcup \bigsqcup \Lambda$ is valid, that is, for any reflexive and transitive interpretation I, there is a $x \in \Delta^I$, such that if $x \in (\bigsqcap \Gamma)^I$, then $x \in (C)^I$ or $x \in (\bigsqcup \Lambda)^I$.

It is then not difficult to conclude that if $x \in (\bigsqcap \Gamma)^I$ and $x \notin (C)^I$, then $x \in (\bigsqcup \Lambda)^I$, that is, $\bigsqcap \Gamma \sqcap \neg C \sqsubseteq \bigsqcup \Lambda$ is also valid.

We proceed analogously for the cases when the last step of the proof are rules for boolean operators.

Consider now the last proof step is the following:

$$\frac{C, \Gamma \Rightarrow \Lambda}{\forall r.C, \Gamma \Rightarrow \Lambda}.$$

By induction we thus know that for any reflexive and transitive interpretation I, the following holds:

$$\left(C \sqcap \bigsqcap \Gamma\right)^I \subseteq \bigsqcup \Lambda^I.$$

It is also known that $(\forall r.C)^I \subseteq (C)^I$ (by Proposition 1), due to reflexivity.

It is then clear that $(\forall r.C \sqcap \bigsqcap \Gamma)^I \subseteq (\bigsqcup \Lambda)^I$.

We consider now the final case:

$$\frac{\forall r.\Gamma \Rightarrow C}{\forall r.\Gamma \Rightarrow \forall r.C}.$$

By induction $\forall r.\Gamma \sqsubseteq C$ is valid.

Now, we know that $\forall r.\forall r.\Gamma \sqsubseteq \forall r.C$ is valid, due to Lemma 1 and that $\forall r.C \sqcap D \equiv \forall r.C \sqcap \forall r.D$ is also valid for any C, D and r.

Since all interpretations are transitive, then we also know that $\forall r.\Gamma \sqsubseteq \forall r.\forall r.\Gamma$ (Proposition 2) is valid.

Therefore, $\forall r.\Gamma \sqsubseteq \forall r.C$ is valid. □

We are going to show completeness by means of ALCS4 axiomatization completeness.

Theorem 4. *If a concept description C is derivable in ALCS4 axiomatization, then there is a proof in ALCS4 sequent system, that is,*

$$\textit{if } \vdash C, \textit{ then } \vdash_G C.$$

Proof. We proceed by induction on derivations in ALCS4 axiomatization. The base cases are the axioms and Modus Ponens. A1 has already been shown in an example above. We now show only A4, A5 and A6, which are the representatives axioms of S4.

— A4:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{C \Rightarrow D, C}\,Ax \quad \dfrac{\dfrac{}{C, D \Rightarrow D}\,Ax}{C \Rightarrow (\neg D), D}\,\neg K}{C \Rightarrow (C \sqcap (\neg D)), D}\,\sqcap K}{(\neg(C \sqcap (\neg D))), C \Rightarrow D}\,\neg A}{(\forall r.(\neg(C \sqcap (\neg D)))), C \Rightarrow D}\,\forall r A}{(\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow D}\,\forall r A}{(\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow (\forall r.D)}\,\forall r K}{(\neg(\forall r.D)), (\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow}\,\neg A}{((\forall r.C) \sqcap (\neg(\forall r.D))), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow}\,\sqcap A}{((\forall r.(\neg(C \sqcap (\neg D)))) \sqcap ((\forall r.C) \sqcap (\neg(\forall r.D)))) \Rightarrow}\,\sqcap A}{\Rightarrow (\neg((\forall r.(\neg(C \sqcap (\neg D)))) \sqcap ((\forall r.C) \sqcap (\neg(\forall r.D)))))}\,\neg K,$$

— A5:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\forall r.C) \Rightarrow (\forall r.C)}\,Ax}{(\forall r.C) \Rightarrow (\forall r.(\forall r.C))}\,\forall r K}{(\neg(\forall r.(\forall r.C))), (\forall r.C) \Rightarrow}\,\neg A}{((\forall r.C) \sqcap (\neg(\forall r.(\forall r.C)))) \Rightarrow}\,\sqcap A}{\Rightarrow (\neg((\forall r.C) \sqcap (\neg(\forall r.(\forall r.C)))))}\,\neg K,$$

— A6:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{C \Rightarrow C}\,Ax}{(\forall r.C) \Rightarrow C}\,\forall r A}{(\neg C), (\forall r.C) \Rightarrow}\,\neg A}{((\forall r.C) \sqcap (\neg C)) \Rightarrow}\,\sqcap A}{\Rightarrow (\neg((\forall r.C) \sqcap (\neg C)))}\,\neg K,$$

— Modus Ponens:

$$\dfrac{\dfrac{\dfrac{}{C \Rightarrow D, C}\,Ax \quad \dfrac{\dfrac{}{C, D \Rightarrow D}\,Ax}{C \Rightarrow (\neg D), D}\,\neg K}{C \Rightarrow (C \sqcap (\neg D)), D}\,\sqcap K}{(\neg(C \sqcap (\neg D))), C \Rightarrow D}\,\neg A \quad .$$

Here $(\neg(C \sqcap (\neg D)))$ corresponds to $C \sqsubseteq D$ by translating subsumption and disjuntion.

For the induction step, we assume we have a proof for a concept description C, then we show the necessitation rule:

$$\dfrac{\dfrac{}{\Rightarrow C}}{\Rightarrow (\forall r.C)}\,\forall r K .$$

□

We are now ready to show completeness, which is immediate from Theorem 4 and Theorem 2.

Corollary 1 (Completeness). *For any reflexive and transitive interpretation I, if I is a model for a concept description C, then there is a proof of C in ALCS4 sequent system, that is, $\vdash_G C$.*

It is important to state the existence of a hierarchy for the application of the rules. We first apply logical rules, followed by the right quantification rule (if possible) and concluding with the left quantification rule.

3.3 Complexity and Implementation

We now show the sequent system for ALCS4 is in EXPTIME. This is mainly due to the fact that proofs are binary tree shaped and there is an exponential bound on the number of binary trees.

Theorem 5 (Complexity). *ALCS4 sequent system is in EXPTIME.*

Proof. First notice that the system satisfies the subformula property, that is, for each rule (S_0, S_1) or (S_0, S_1, S_2), formulas in S_1 or S_2 are all subformulas of the ones occurring in S_0.

It is then clear that the height of the proofs (number of proof steps) is linear with respect to the size of the input sequents.

Now notice that due to the conjunction on the right rule, proofs are binary tree shaped.

The exponential bound comes from the fact that the number of nodes in binary trees are exponentially bounded with respect to height of the tree. □

Regarding implementation, we followed a functional approach as in [13], more precisely, the ALCS4 system was implemented in ML [8].

We consider three fundamental data types: concept descriptions, whose alphabet is formed by concept names and roles; sequents, defined as a pair of concept lists; and rules, which can be axioms, and rules with one or two hypothesis. They are depicted in Figure 2.

The code for the implementation of rules described in Definition 6 is depicted in Figure 3.

```
datatype Roles=  role of string;

datatype Conc= conc of string |
               nega of Conc |
               conj of Conc * Conc|
               cuan of Roles * Conc;

type Seq = Conc list;

datatype Sequent = lr of Seq * Seq;

datatype SystemG = Ax of Sequent |
InRuOne of Sequent |
InRuTwo of Sequent * Sequent;
```

Fig. 2. Data Types in ML for ALCS4 sequent system

The entire code for the implementation is in `http://aleteya.cs.buap.mx/~iebp/ALCS4_Fin.ML`, which was tested in PC with Intel Core i5 1.8 GHz, Windows 8.1, in the Moscow ML version 2.01.

The proof of axiom A5, which is part of the proof of Theorem 4, takes 16.4844 milliseconds in our implementation.

```
fun reduce2tex1(lr(nega(a)::gamma,delta)) =
InRuOne(lr(gamma,a::delta))
|   reduce2tex1(lr(conj(a,b)::gamma,delta)) =
InRuOne(lr(a::b::gamma,delta))
|   reduce2tex1(lr(gamma,nega(a)::delta)) =
InRuOne(lr(a::gamma,delta))
|   reduce2tex1(lr(gamma,conj(a,b)::delta)) =
InRuTwo(lr(gamma,a::delta),lr(gamma,b::delta))
|   reduce2tex1(lr(gamma,cuan(c)::nil)) =
if ForallIn(r,gamma)
then InRuOne(lr(gamma,c::nil))
else Ax(lr(gamma,cuan(c)::nil))
|   reduce2tex1(lr(cuan(a)::gamma,delta)) =
InRuOne(lr(a::gamma,delta))
|   reduce2tex1(lr(gamma,cuan(c)::rest)) =
if HaveNeg(rest)
then  InRuOne(lr(gamma,cuan(c)::rest))
else
(Ax(lr(gamma,cuan(c)::rest)))
|   reduce2tex1(any) = (decide(any));
```

Fig. 3. ML implementation of ALCS4 sequent system

4 Tree Hypersequents for ALC

In this Section, we first introduce the notion of tree hypersequents, which is a generalization of sequents. Then a tree hypersequent system for ALC is introduced, without any restriction to roles. Proofs for correctness and complexity (2EXPTIME) are also given. The corresponding implementation is also described.

4.1 The Calculus

Tree hypersequents as their name suggest is a tree structure, where each of its nodes is a sequent. The following is a more precise definition.

Definition 8 (Tree hypersequents)**.** Tree hypersequents (THS) are inductively defined by the following grammar:

$$T := S/MT$$
$$MT := \emptyset \mid T; MT,$$

where S is a sequent.

Regarding notation, instead of $S/\emptyset$, we write S.

As in [10], we now define the zoom for tree hypersequents.

Definition 9. The set of zoom tree hypersequents ($ZTHS$) is inductively defined in the following way:

- $[-] \in ZTHS$.
- If $T_1, ..., T_n \in THS$, $r_1, ..., r_n$ are roles names, then $[-]/r_1 : T_1; ...; r_n : T_n \in ZTHS$.
- If S is a sequent, $T_2, ..., T_n \in THS$, $r_1, ..., r_n$ are roles names and $T_1[-] \in ZTHS$, then $S/r_1 : T_1[-]; r_2 : T_2; ...; r_n : T_n \in ZTHS$.

$T[T']$ denote the substitution of T' in $T[-]$ which is defined as follows:

- If $T[-] = [-]$, then $T[T'] = T'$.
- If $T[-] = [-]/r_1 : T_1; ...; r_n : T_n$ and $T' = S'/w_1 : T'_1; ...; w_k : T'_k$, then $T[T'] = S'/r_1 : T_1; ...; r_n : T_n; w_1 : T'_1; ...; w_k : T'_k$.
- If $T[-] = S/r_1 : T_1[-]; r_2 : T_2; ...; r_n : T_n$, then $T[T'] = S/r_1 : T_1[T']; r_2 : T_2; ...; r_n : T_n$.

Definition 10 (ALC tree hypersequent system)**.** We define a tree hypersequent system for ALC by the following rules.

$$\frac{}{T[C, \Gamma \Rightarrow C, \Lambda]}\, Ax,$$

$$\frac{T[\Gamma \Rightarrow C, \Lambda]}{T[\neg C, \Gamma \Rightarrow \Lambda]}\, \neg A,$$

$$\frac{T[C, \Gamma \Rightarrow \Lambda]}{T[\Gamma \Rightarrow \neg C, \Lambda]}\, \neg K,$$

$$\frac{T[C, D, \Gamma \Rightarrow \Lambda]}{T[C \sqcap D, \Gamma \Rightarrow \Lambda]}\, \sqcap A,$$

$$\frac{T[\Gamma \Rightarrow C, \Lambda] \quad T[\Gamma \Rightarrow D, \Lambda]}{T[\Gamma \Rightarrow C \sqcap D, \Lambda]}\, \sqcap K,$$

$$\frac{T\,[(\forall r.C, \Gamma \Rightarrow \Lambda)/r : (C, \Gamma' \Rightarrow \Lambda'/MT'); MT]}{T\,[(\forall r.C, \Gamma \Rightarrow \Lambda)/r : (\Gamma' \Rightarrow \Lambda'/MT'); MT],}\, \forall r A,$$

$$\frac{T[(\Gamma \Rightarrow \Lambda)/r(\Rightarrow C)]}{T[\Gamma \Rightarrow \forall r.C, \Lambda],}\, \forall r K.$$

Notice that the $\forall rA$ rule can be applied to an hypersequent $T[(\forall r.C, \Gamma \Rightarrow \Lambda)/MT]$ provided that there exists an hypersequent in MT that is labeled by the r role. Hence, if every hypersequent in M labeled by r contains the concept C in the antecedent of the root, then the rule $\forall rA$ can not be applied and we can thus no longer consider the concept $\forall r.C$.

As an example of the use of the tree hypersequent system, consider the following proof of A4:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Rightarrow /r : (C \Rightarrow D, C)}\,Ax \quad \dfrac{\dfrac{}{\Rightarrow /r : (D, C \Rightarrow D)}\,Ax}{\Rightarrow /r : (C \Rightarrow (\neg D), D)}\,\neg K}{\Rightarrow /r : (C \Rightarrow (C \sqcap (\neg D)), D)}\,\sqcap A}{\Rightarrow /r : ((\neg(C \sqcap (\neg D))), C \Rightarrow D)}\,\neg A}{(\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow /r : (C \Rightarrow D)}\,\forall r A}{(\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow /r : (\Rightarrow D)}\,\forall r A}{(\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow (\forall r.D)}\,\forall r K}{(\neg(\forall r.D)), (\forall r.C), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow}\,\neg A}{((\forall r.C) \sqcap (\neg(\forall r.D))), (\forall r.(\neg(C \sqcap (\neg D)))) \Rightarrow}\,\sqcap A}{((\forall r.(\neg(C \sqcap (\neg D)))) \sqcap ((\forall r.C) \sqcap (\neg(\forall r.D)))) \Rightarrow}\,\sqcap A}{\Rightarrow (\neg((\forall r.(\neg(C \sqcap (\neg D)))) \sqcap ((\forall r.C) \sqcap (\neg(\forall r.D)))))}\,\neg K.$$

4.2 Correctness

We now show the ALC tree hypersequent system is correct.

Tree hypersequents are interpreted as a disjunction composed by its branches. More precisely, we inductively define the following interpretation function:

— $(C_1, \ldots, C_n \Rightarrow D_1, \ldots, D_m)^H = \bigsqcap_{i=1}^{n} C_i \sqsubseteq \bigsqcup_{i=1}^{m} D_i$, and

— $(S/r_1 : T_1, \ldots, r_k : T_k)^H = S^H \sqcup \bigsqcup_{i=1}^{k} \forall r_i.({T_i}^H)$.

Theorem 6 (Soundness)**.** *If there is a proof of the tree hypersequent T in the ALC tree hypersequent system, then T^H is valid.*

Proof. We proceed by induction on the proof height of T.

In the base case, we distinguish two cases: one when $C, \Gamma \Rightarrow C, \Lambda$ occurs at the root of T, and the other when it does not.

In the first case it is clear that $C \sqcap \bigsqcap \Gamma \sqsubseteq C \sqcup \bigsqcup \Lambda$ is valid.

In the second case, it is also not difficult to see the validity of $\forall r_1. \ldots .\forall r_n.(C \sqcap \bigsqcap \Gamma \sqsubseteq C \sqcup \bigsqcup \Lambda)$, such that a concept D is satisfied, then $\forall r D$ is also satisfied for any role r, given that $\Delta^I \subseteq D^I$ is complied.

In the induction step we assume a proof of a tree T', and then we show by cases the rule $\frac{T'}{T}$ can be applied.

Consider now the case when the last proof step is the following:

$$\frac{T[\Gamma \Rightarrow C, \Lambda]}{T[\neg C, \Gamma \Rightarrow \Lambda].}$$

In this case, we also distinguish two subcases: when $\neg C, \Gamma \Rightarrow \Lambda$ occurs as the root of T, and when it does not.

In the first case, we know that $\Gamma \Rightarrow C, \Lambda$ occurs also in the root of T and it is also valid (inductive hypothesis). Then for any interpretation I, we know that $(\bigsqcap \Gamma)^I \subseteq C^I \cup (\bigsqcup \Lambda)^I$. From this, we imply $(\bigsqcap \Gamma)^I \cap (C^I)^c \subseteq (\bigsqcup \Lambda)^I \cap (C^I)^c \subseteq (\bigsqcup \Lambda)^I$, and hence $\neg C, \Gamma \Rightarrow \Lambda$ is valid.

The other cases when the last proof step involves boolean operators are proven in an analogous manner.

We focus now on the case of the quantification on the left.

$$\frac{T\left[(\forall r.C, \Gamma \Rightarrow \Lambda)/r : (C, \Gamma' \Rightarrow \Lambda'/MT')\right]}{T\left[(\forall r.C, \Gamma \Rightarrow \Lambda)/r : (\Gamma' \Rightarrow \Lambda'/MT')\right]} \quad .$$

We only prove the basic case and when $MT' = \emptyset$, the arguments for the other cases are similar. Then, by inductive hypothesis, we obtain that the following concept is valid.

$$\neg\left(\forall r.C \sqcap \bigsqcap \Gamma\right) \sqcup \bigsqcup \Lambda$$
$$\sqcup \forall r.\left(\neg\left(C \sqcap \bigsqcap \Gamma'\right) \sqcup \bigsqcup \Lambda'\right).$$

Then, by De Morgan's laws:

$$\neg\forall r.C \sqcup \neg \bigsqcap \Gamma \sqcup \bigsqcup \Lambda \sqcup \forall r.\left(\neg C \sqcup \neg \bigsqcap \Gamma' \sqcup \bigsqcup \Lambda'\right).$$

And by normality:

$$\neg\forall r.C \sqcup \neg \bigsqcap \Gamma \sqcup \bigsqcup \Lambda \sqcup$$
$$\forall r.\neg C \sqcup \forall r.\left(\neg \bigsqcap \Gamma' \sqcup \bigsqcup \Lambda'\right)$$

and since $\forall r.\neg C \sqsubseteq \neg\forall r.C$ is satisfied, then it clearly implies that $((\forall r.C, \Gamma \Rightarrow \Lambda)/r : (\Gamma' \Rightarrow \Lambda'))^H$ is valid.

Quantification on the right:

$$\frac{T[(\Gamma \Rightarrow \Lambda)/r(\Rightarrow C)]}{T[\Gamma \Rightarrow \forall r.C, \Lambda]} \quad .$$

We again only prove the base case when $T[-] = [-]$. By induction, we know that the following concept is valid $(\neg \bigsqcap \Gamma) \sqcup \bigsqcup \Lambda \sqcup \forall r.(\neg\top \sqcup C)$. Then $(\neg \bigsqcap \Gamma) \sqcup \bigsqcup(\forall r.C, \Lambda)$ is valid which clearly implies that the root hypersequent is valid under the function H. □

Completeness is also proven with respect to ALC axiomatization completeness. We then first show the ALC tree hypersequent system is complete with respect to ALC axiomatization.

Theorem 7. *If a concept description C is derivable in ALC axiomatization, then there is a proof in the ALC tree hypersequent system, that is,*

$$\textit{if } \vdash C, \textit{ then } \vdash_H C.$$

Proof. The proof goes by induction on derivations in ALC axiomatization. In the base case, the only interesting subcase is the proof of axiom A4, which has been given in the example above. For the induction step, we assume we already have a proof of a concept C, then we prove the necessitation rule.

$$\frac{\overline{\Rightarrow /r : (\Rightarrow C)}}{\Rightarrow (\forall r.C)} \; \forall r K \; .$$

□

Completeness is now clear from Theorem 7 and Theorem 1.

Corollary 2 (Completeness)**.** *For any interpretation I, if I is a model for a concept description C, then there is a proof of C in the ALC tree hypersequent system, that is, $\vdash_H C$.*

4.3 Complexity and Implementation

The complexity of the tree hypersequent system for ALC is in 2EXPTIME. Intuitively, proofs in this system are binary trees where the number of nodes of a tree is exponentially bounded by its height, which is itself exponentially bounded by the size of the input concept.

Theorem 8 (Complexity)**.** *The ALC tree hypersequent system is in 2EXPTIME.*

Proof. The first exponential bound comes from the bound on the number of nodes in binary trees with respect to the tree height. Recall from the complexity proof of the ALCS4 sequent system that rules corresponding to boolean connectives produce binary shaped proof trees, due to the subformula property.

We now show the second exponential bound: the height of proof trees is exponentially bounded by the size of the input concept.

For this purpose, first recall that each node of the proof tree is a tree hypersequent. We then proceed to show that the number of nodes for each tree hypersequent is exponentially bounded by the concept size.

First, due to the subformula property, it is easy to see that the rank of each tree hypersequent is linearly bounded by the concept size. However, due to the universal quantiflcation on the left rule, there are duplications (nodes in the tree hypersequents).

Now, there is a well-known and straightforward linear reduction of n-ary trees to binary trees [2, 3, 4], such that the height of the binary trees is linearly bounded by the rank of the n-ary trees.

Then, the height of the binary version of the tree hypersequents is also linear.

Recalling again the exponential bound on the number of nodes in binary trees with respect to the height, we obtain that the size of each tree hypersequent (proof node) is exponential with respect to the input concept size. □

The implementation of ALC tree hypersequent system is similar to the one described for the ALCS4 sequent system in Section 3. However, in the case of the ALC system, the fundamental data structure is composed by tree hypersequents, which is described in Figure 4. In this Figure, a data type for proof trees with tree hypersequents as nodes is also displayed. This data type may have no applicable rule, an axiom, or a rule with one or two hypothesis. The proof of axiom A4, depicted above, takes 19.0497 milliseconds in our implementation. The full code is described in `http://aleteya.cs.buap.mx/~iebp/ALC_Fin.ML`.

```
datatype THS =
        hyper of Sequent *
        ((Roles * THS) list);

datatype SystemGH = NoRule
 | AXITHS of THS
 | InRuOneH of THS
 | InRuTwoH of THS * THS;
```

Fig. 4. Data structures for tree hypersequent proofs

5 Conclusions

In this paper, we introduce the proof theoretic basics for two important description logics (DL): the basic propositionally closed concept language ALC and its variant with reflexive and transitive roles, which we name ALCS4 due to the straightforward correspondence with the multi-modal logic S4. The system for ALCS4 enjoys of cut elimination, which allows to prove decidability. In addition, this system is also contraction-free. These two features are important requirements for the implementation of sequent-like systems. An implementation is also provided. The system is also proven correct: sound and complete. And the complexity of the system is shown to be in EXPTIME.

We also introduce a proof system for ALC without restrictions on roles. The system is based on tree hypersequents, which is generalization of sequents. This system is also cut and contraction free. Correctness and complexity (2EXPTIME) proofs of the system are also given together with an implementation.

Since both systems described in this work constructively search for candidate proofs, counter-models maybe extracted from unsuccessful proofs. We are currently developing the corresponding algorithms.

We believe the reasoning frameworks provided in the current work will allow further studies in description logics from a proof theoretic perspective. In particular, we are interested in constructive proofs of Craig interpolation in the DL context.

Another immediate research perspective is the study of sequent-like proof system for description logics including terminological (TBoxes) and assertional (ABoxes) reasoning.

Also in this setting, we plan to study more expressive description logics, including arithmetical (numerical) restrictions, inverse roles and nominals [2, 3, 4].

Acknowledgment

This work was partially developed under the support of the Mexican National Science Council (CONACYT) in the scope of the Cátedras CONACYT project Infraestructura para Agilizar el Desarrollo de Sistemas Centrados en el Usuario (Ref. 3053).

ISSN 2007-9737

References

1. **Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F.**, editors (**2003**). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

2. **Bárcenas, E., Genevès, P., Layaïda, N., & Schmitt, A.** (**2011**). Query reasoning on trees with types, interleaving, and counting. **Walsh, T.**, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, IJCAI/AAAI, pp. 718–723.

3. **Bárcenas, E. & Lavalle, J.** (**2013**). Expressive reasoning on tree structures: Recursion, inverse programs, Presburger constraints and nominals. **Espinoza, F. C., Gelbukh, A. F., & González-Mendoza, M.**, editors, *12th Mexican International Conference on Artificial Intelligence*, volume 8265 of *Lecture Notes in Computer Science*, Springer, pp. 80–91.

4. **Bárcenas, E. & Lavalle, J.** (**2014**). Global numerical constraints on trees. *Logical Methods in Computer Science*, Vol. 10, No. 2.

5. **Blackburn, P., van Benthem, J., & Wolter, F.** (**2006**). *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA.

6. **De Giacomo, G. & Lenzerini, M.** (**1994**). Boosting the correspondence between description logics and propositional dynamic logics. **Hayes-Roth, B. & Korf, R. E.**, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Volume 1.*, AAAI Press / The MIT Press, pp. 205–212.

7. **Gentzen, G.** (**1935**). Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, Vol. 39, pp. 176–210, 405–431.

8. **Gordon, M. J. C., Milner, R., Morris, L., Newey, M. C., & Wadsworth, C. P.** (**1978**). A metalanguage for interactive proof in LCF. **Aho, A. V., Zilles, S. N., & Szymanski, T. G.**, editors, *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, ACM Press, pp. 119–130.

9. **Goré, R. & Ramanayake, R.** (**2012**). Labelled tree sequents, tree hypersequents and nested (deep) sequents. **Bolander, T., Braüner, T., Ghilardi, S., & Moss, L. S.**, editors, *Advances in Modal Logic 9*, College Publications, pp. 279–299.

10. **Hill, B. & Poggiolesi, F.** (**2010**). A contraction-free and cut-free sequent calculus for propositional dynamic logic. *Studia Logica*, Vol. 94, No. 1, pp. 47–72.

11. **Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S.**, editors (**2009**). *OWL 2 Web Ontology Language: Primer*. W3C Recommendation.

12. **Horrocks, I., Sattler, U., & Tobies, S.** (**1999**). Practical reasoning for expressive description logics. **Ganzinger, H., McAllester, D. A., & Voronkov, A.**, editors, *Logic Programming and Automated Reasoning, 6th International Conference*, volume 1705 of *Lecture Notes in Computer Science*, Springer, pp. 161–180.

13. **Muñoz-Toriz, J.-P., Martínez-Ruiz, I., & Arrazola-Ramírez, J.** (**2014**). On automatic theorem proving with ML. *13th Mexican International Conference on Artificial Intelligence*, IEEE.

14. **Negri, S.** (**2011**). Proof theory for modal logic. *Philosophy Compass*, Vol. 6, No. 8, pp. 523–538.

15. **Poggiolesi, F.** (**2008**). *Sequent Calculi for Modal Logic*. Ph.D. thesis, University of Florence.

16. **Rademaker, A.** (**2012**). *A Proof Theory for Description Logics*. Springer Briefs in Computer Science. Springer.

17. **Schild, K.** (**1991**). A correspondence theory for terminological logics: Preliminary report. **Mylopoulos, J. & Reiter, R.**, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence.*, Morgan Kaufmann, pp. 466–471.

18. **Schmidt-Schauß, M. & Smolka, G. (1991)**. Attributive concept descriptions with complements. *Artif. Intell.*, Vol. 48, No. 1, pp. 1–26.

19. **Sun, Y. & Sui, Y. (2014)**. A Gentzen system for the description logic. *9th International Conference on Computer Science and Education*, pp. 99–102.

Juan Pablo Muñoz is a doctoral student in Mathematical Sciences at the Benemérita Universidad Autónoma de Puebla and also he is a grant holder by the Consejo Nacional de Ciencia y Tecnología (CONACYT). His main research areas are non-classical logic, proof theories, automated theorem proving, and knowledge representation.

Everardo Bárcenas is a researcher at the Consejo Nacional de Ciencia y Tecnología (CONACYT). He graduated from Université de Grenoble. His research interests include automated reasoning, modal logics, knowledge representation, and formal methods.

Iván Martínez is a full-time professor of the Facultad de Ciencias Físico-Matemáticas at BUAP. He obtained his Ph.D. in Mathematical Sciences from the National Autonomus University of Mexico (UNAM) in 2010. His research interests are mathematical logic, set theory, logic programming, knowledge representation, and general topology.

José Ramón Enrique Arrazola received his Bachelor's, Master's and Doctorate degrees from Benemérita Universidad Autónoma de Puebla in Puebla, Mexico, in 1986, 1989 and 1996, respectively. He has directed over 13 undergraduate theses, nine Master's theses, and one doctoral thesis. Also, he has around 70 articles published. His research interests are mathematical logic, logic programming and knowledge representation, argumentation theory and topology. He is the current Director of the Physical-Mathematical Sciences Faculty at the Benemérita Universidad Autónoma de Puebla.

Article received on 30/01/2015, accepted on 28/07/2015.
Corresponding author is Juan Pablo Muñoz.