I
Introduction

I.1 Description Logics

Description Logics (**DLs**) are quite well-established as underlying logics for Knowledge Representation (**KR**). Part of this success come from the fact that it can be seem as one (logical) successor of Semantics Networks [52], Frames [48] and Conceptual Graphs [65] and as well as, an elegant and powerful restriction of **FOL** by guarded prefixes, that also leads to a straight interpretation into the **K** propositional modal logic.

The core of the **DLs** is the $\mathcal{ALC}$ description logic. In a broader sense, a Knowledge Base (**KB**) specified in any description logic having $\mathcal{ALC}$ as core is called an Ontology. In this thesis we will not take any ontological[1] discussion on the choice for this terminology by the computer science community. Moverover, we are not interested in the technological concerns around Ontologies, the **Web** or the fact that there is a XML dialect for writing Ontologies, just named OWL [32]. For us, a DL theory presentation, that is, a set of axioms in the **DL** logical language, and, an OWL file containing the same set of axioms is the same **KB**.

Description Logics is a family of formalisms used to represent knowledge of a domain. In contrast with others knowledge representation systems, Description Logics are equipped with a formal, logic-based semantics. This logic-based semantics provides to systems based on it various inference capabilities to deduce implicit knowledge from the explicitly represented knowledge.

I.2 Motivation

Research in DL, since the beginning, was oriented to the development of systems and to their use in applications. In the first half of the 1980's several systems were developed including KL-ONE [11] and KRYPTON [10], only to mention two. They were called first generation DL systems. Later, in the second

[1] In the philosophical sense.

half of 1980's, the second generation of DL systems appear, the BACK [37], CLASSIC [9] and LOOM [41] systems.

In the last years several DL systems have been developed incorporating different DL fragments but similar with respect to the underlying reasoning algorithm. Nowadays, DL has good reasoners from the point of view of providing yes/no answers or various inference tasks like subsumption of concepts (see Chapter II) or classification [2]. We mention the open-source Pellet [63], Racer Pro [33] and Fact [67]. [3]

The first DL systems implement structural subsumption algorithms [61]. The basic idea underlying structural subsumption is to transform terms into *canonical normal forms*, which are then structurally compared. Structural subsumption algorithms are therefore also referred to as normalize-compare algorithms. There is one important drawback of normalize-compare algorithms. That is, in general it is straightforward to prove the correctness of such algorithms but there is no method for proving their completeness [51].

As far as we know, the most well-known existing DL reasoners implement variations of Tableaux proof procedure for DL [60, 24, 23]. As pointed in [51], Tableaux procedures for computing subsumption of concepts had the advantage of providing good basis for theoretical investigations. Not only was their correctness and completeness easy to prove, they also allowed a systematic study of the decidability and the tractability of different DL dialects. On the other hand, the main disadvantage of tableaux-based algorithms is that they are not constructive but rather employ refutation techniques. That is, in order to prove $\alpha \sqsubseteq \beta$, it is proved that the concept $\alpha \sqcap \neg\beta$ is not satisfiable (see Chapter II).

As claimed by [43], the use of Description Logics by regular users, that is, non-technical users, would be wider if the computed inferences could be presented as a natural language text – or any other presentation format at the domain's specification level of abstraction – without requiring any knowledge on logic to be understandable.

Despite the higher efficiency of the recent available DL systems they do not provide to ontology engineers a good support for explanations on their two main uses, namely, answering whether a subsumption holds or not, and, a classification result.

Some works ([43, 44, 40]) describe methods to extract explanations from DL-Tableaux proofs. Particularly, [21] describes the explanation extraction

[2] The classification checks subsumption between the terms defined in the terminology and computes the subsumption hierarchy of them.

[3] A possible outdated list is maintained in the Description Logics website `http://dl.kr.org/`.

in quite few details, making impossible a feasible comparison with [44, 40]. In [7], for example, it is described a Sequent Calculus (SC) obtained by a standard transformation from Tableaux into SC systems applied to the DL-tableaux described in [60]. [38] presents also a Resolution procedure for DL but does not address explanation extraction. It is worth noticing that the DL-tableaux do not implement non-analytic cuts, and hence proof resulted from this transformation is a cut-free proof. Moreover, even when dealing with the **TBOX** (see Chapter II) the SC just discussed strongly deals with individuals, the **ABOX** aspect of an Ontology.

Simple Tableaux procedure are those not able to implement non-analytic cuts. The Tableaux procedures used for $\mathcal{ALC}$ are simple. It is also known that Simple Tableaux cannot produce always short proofs, [4] that is, polynomially lengthy proofs, concerning the combined length[5] of its conclusion and set of (used) axioms from the Ontology. This is an easy corollary of the theorem that asserts that Simple Tableaux as well Resolution cannot produce short proofs for the Pigeonhole Principle (PHP) [36]. PHP is easily expressed in propositional logic, and hence, is also easily expressed in $\mathcal{ALC}$. On the other hand, Sequent Calculus (SC) (with the cut rule) has short proofs for PHP. In [31, 27] it is shown, distinct, SC proof procedures that incorporate mechanisms that are somehow equivalent to the cut-rule. Anyway, both articles show how to obtain short proof, in SC, for the PHP. We believe that super-polynomial proofs, like the ones generated by simple Tableaux, cannot be considered as good sources for text generation. The reader might want to consider that only the reading of the proof itself is a super-polynomial task regarding time complexity.

The final consideration worth of mentioning regarding a motivation to obtain a Natural Deduction system for $\mathcal{ALC}$, despite providing a variation of themes, is the possibility of getting ride on a weak form of the Curry-Howard isomorphism in order to provide explanations with greater content. This last affirmative takes into account that the reading (explanatory) content of a proof is a direct consequence of its computational content. This is discussed in Chapters V and VII.

A last observation lies on the fact that allowing this incremental proof-theoretical design of systems to **DL** we obtain a uniform specialization of the general proof procedure for $\mathrm{ND}_{\mathcal{ALC}}$.

[4] If we consider the assumption that $NP \neq CONP$.
[5] Number of symbols.

I.3 What this thesis is about

In this thesis, we present two deduction systems for $\mathcal{ALC}$ [6] and $\mathcal{ALCQI}$ [7], a sequent calculus and a natural deduction system. The first motivation for developing such systems is the extraction of computational content of $\mathcal{ALC}$ and $\mathcal{ALCQI}$ proofs. More precisely, these systems were developed to allow the use of natural language to render a Natural Deduction proof. The sequent calculi were intermediate steps towards a Natural Deduction Systems [19].

Our main motivation to develop such systems are that natural language rendering of a Natural Deduction proofs is worthwhile in a context like proof of conformance in security standards [22]. The research reported on this thesis started in the context of a joint project between PUC–Rio TecMF Lab and Modulo Security S.A.

Our Sequent Calculus is also compared with other approaches like Tableaux [60] and the Sequent Calculus for $\mathcal{ALC}$ [28, 45, 6] based on this very Tableaux. In fact, our system does not use individual variables (first-order ones) at all. The main mechanism in our system is based on labeled formulas. The labeling of formulas is among one of the most successful artifacts for keeping control of the context in the many existent quantification in logical system and modalities. For a detailed reading on this approach, we point out [56, 35, 57, 58, 29].

Our Sequent Calculi systems argue in favour of better explanation schemata obtained from proofs, regarding those obtained from a $\mathcal{ALC}$-Tableaux. Both systems do not use individuals, producing a purely conceptual reasoning for TBOX. Moreover, it is worth of mentioning that both systems can also provide proofs with cuts, as opposed to the one presented in [43].

I.4 How this thesis is organized

Chapter II presents some background introducing DL languages and semantics.

Chapter III presents the system $\mathrm{SC}_{\mathcal{ALC}}$, a sequent calculus for $\mathcal{ALC}$ and proves that it is sound and complete. This chapter was originally published in [53] and [55], where we proved that $\mathrm{SC}_{\mathcal{ALC}}$ has the desirable property of allowing the construction of cut-free proofs. That is, we prove that the *cut rule* can be eliminated from the system $\mathrm{SC}_{\mathcal{ALC}}$ without lost the completeness and soundness.

[6] $\mathcal{ALC}$ means Attributive Language with Complements, a basic Description Logic.

[7] The $\mathcal{Q}$ in $\mathcal{ALCQ}$ means the introduction in the language of qualified number restriction constructors.

In Chapter IV, we compare $\mathrm{SC}_{\mathcal{ALC}}$ with the Structural Subsumption algorithm and the Tableaux for $\mathcal{ALC}$. The comparison is made regarding: (1) the proof construction procedure in Structural Subsumption algorithm and $\mathrm{SC}_{\mathcal{ALC}}$; and (2) the ability of $\mathcal{ALC}$-Tableaux to construct counter-models. The results from this chapter were first published in [54].

In Chapter V we present the Natural Deduction for $\mathcal{ALC}$ named $\mathrm{ND}_{\mathcal{ALC}}$. In this chapter, we also prove that $\mathrm{ND}_{\mathcal{ALC}}$ is sound and complete. We also proof the normalization theorem for $\mathrm{ND}_{\mathcal{ALC}}$. The results in this chapter were published in [34].

In Chapter VI, we present the extensions of our Natural Deduction and Sequent Calculus for $\mathcal{ALC}$ to $\mathcal{ALCQI}$. We prove the soundness of both systems and some ongoing work regarding their completeness. In Chapter VII, we present the motivation and some discussion about the extraction of explanations from proofs. We compare proofs in Tableaux, Natural Deduction and Sequent Calculi. Also in this chapter, we present our Natural Deduction for $\mathcal{ACLQI}$ to reasoning over a UML diagram. The example helps us compare how proofs in $\mathrm{ND}_{\mathcal{ALCQI}}$ can be easier explain than proofs using Tableaux.

In Chapter VIII, we present a prototype theorem prover that implements our Natural Deduction and Sequent Calculi systems. Finally, in Chapter IX, we present some conclusions and further work.