A Simple Parallel Reasoning System for the \textit{ALC} Description Logic

Adam Meissner

Institute of Control and Information Engineering, Poznań University of Technology, pl. M. Skłodowskiej-Curie 5, 60-965 Poznań, Poland
Adam.Meissner@put.poznan.pl

Abstract. In this paper we present a simple, tableau-based, parallel reasoning system for the \textit{ALC} description logic. The system is built in relational model in the Oz language and has a form of a short program comprising the implementation of tableau rules. The program can be executed according to various strategies, particularly in parallel on distributed machines. For this purpose, we use a parallel search engine available in the Mozart environment. We describe results of experiments for estimating the speedup obtained by parallel processing.

Keywords: parallel reasoning, \textit{ALC} description logic, Oz language.

1 Introduction

The last fifteen years is a period of a growing interests in Description Logics (DLs) [3]. This term denotes a wide and rather eclectic group of formal systems mainly intended for representing and processing a terminological knowledge. Description Logics can be classified by languages they support. A particular attention is paid to a family of DLs with the \textit{ALC} language as a core formalism (so called \textit{ALC} DLs). Members of this class show a reasonable expressivity and, in many cases, are decidable. This makes them attractive for knowledge engineers – \textit{ALC} DLs are successively applied in various fields, for example in software engineering, object data bases, medical expert systems, control in manufacturing, action planning in robotics and, for last eight years, in Semantic Web [8].

One of the most important issues in DL research is the development of efficient reasoning methods and systems since the majority of inference problems occurring in this area can not be solved in polynomial time. In particular, a time of finding a solution for a given problem can be reduced by parallelizing the inference process. This direction of research in the domain of automated theorem provers (ATPs) has been carrying out since late eighties and it yields many theoretical results [2]. However, bringing up these ideas into practice still causes difficulties. A source of significant problems is often the fact that the method by which the computations in ATP are parallelized is a part of the system execution strategy. The strategy, in turn, is either hardwired in the system if it is implemented by means of imperative programming or it is a fixed part of a runtime platform (as for instance, in the Prolog language). This blurs the system
architecture, makes it harder to understand and hence increases the probability of error occurrence. Also, it usually causes problems with the scalability of the environment where the given ATP runs.

In this paper we overcome the problems mentioned above by separating the declarative part of the reasoning system from its execution strategy. More precisely, we take a classical tableau-based inference algorithm for $\mathcal{ALC}$ DL and express it in terms of the relational model in the Oz language. The reasoning procedure consists of the implementation of inference rules only while the execution strategy is implemented as the search engine, to wit a special object which runs the given relational program. This makes possible to perform the computations in various ways practically without any modification of the reasoning procedure. Particularly, the reasoning process can run in parallel on distributed machines. For this purpose, we use a parallel search engine available in the Mozart system being a programming environment for the Oz language. It should be noted, that the presented approach can be regarded as an example of lean deduction, which assumes achieving maximal efficiency for minimal means. This idea, including its advantages and limitations, is widely discussed in [1].

The paper is organized as follows. Section 2 contains the principles of the reasoning method, which comprise an outline of $\mathcal{ALC}$ DL tableau calculus and the description of how to express it in the relational model in the Oz programming language. In section 3 we discuss the results of experiments intended for estimating the speedup obtained by parallel computations. Section 4 contains some final remarks.

2 Reasoning Method

Firstly, we outline the syntax and the semantics of the $\mathcal{ALC}$ description logic. Then, we briefly present the basic inference problem for $\mathcal{ALC}$ DL, namely testing for concept satisfiability. We also cite the classical tableau-based algorithm, which solves this problem. Finally, we show how the algorithm can be parallelized by defining it in terms of the relational model in the Oz language.

The elementary expressions in $\mathcal{ALC}$ DL are atomic descriptions (or synonymously, names) of concepts and roles. A concept is a set of individuals, which are called instances of this concept. A role is a binary relation holding between individuals. Any element of a role is called an instance of this role. Concepts, besides names, can also be represented by complex descriptions, which are built from simpler descriptions and special symbols called concept constructors. We use the letter $A$ to denote a concept name and letters $C$ or $D$ as symbols of any concept descriptions; the letter $R$ stands for a role description. All these symbols can possibly be subscripted. The set of $\mathcal{ALC}$ DL concept constructors comprises five elements, namely negation ($\neg C$), intersection ($C \cap D$), union ($C \cup D$), existential quantification ($\exists R.C$) and value restrictions ($\forall R.C$); expressions written in parentheses are schemes of relevant concept descriptions.

In the sequel, if it does not lead to misunderstanding, we often identify descriptions with their meanings (e.g. we say “a concept” instead of “a concept