Chapter 1

Introduction

In this chapter, we introduce the notion of knowledge representation. We emphasize the importance of this concepts and describe its development through the years. Later, we distinguish the types of knowledge. In order to suggest a language for knowledge representation, we review the history of programming languages. We introduce the idea of commonsense reasoning and again present the different types of reasoning. We illustrate an example of the use of commonsense reasoning. Finally, we define our problem to solve.

1.1 Knowledge representation

In this section, we describe the ideas of McCarthy from 1959 about knowledge and types of knowledge. In addition, we briefly mention the development of programming languages through the history.

1.1.1 The ideas of McCarthy

Representing knowledge is an important part of Artificial Intelligence. If we want to create an intelligent agent who is going to act in some environment, we need first to give the agent enough knowledge about that environment. After, we would like the agent to reason and communicate about its knowledge and beliefs. Therefore, we need a language to represent the knowledge and also rules that help manipulate that knowledge [Baral and Gelfond, 1994]. Let us first review the beginning of knowledge representation.

The notion of formalizing knowledge is relatively new. John McCarthy was the first scientist to talk about commonsense reasoning. In his paper of 1959, he explains that *a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows* [McCarthy, 1959]. In addition, McCarthy mentions that an intelligent system should have at least 5 basic characteristics:

- All behaviors must be representable in the system. That is, the richness of language is important.
- Interesting changes in behavior must be expressible in a simple way. In other words, the elaboration tolerance property must hold (add new relevant information without major changes in the previous one).
- All aspects of behavior except the most routine must be improvable. This means that the heuristics can be modified on the fly.
- The machine must have or evolve concepts of partial success. For instance, anytime algorithms can achieve this goal, because they return the best answer possible even if it is not allowed to run to completion, and may improve on the answer if it is allowed to run longer.
- The system must be able to create subroutines (a group of instructions that perform a specific task), which can be included in procedures as units.

In [McCarthy, 1969] McCarthy gives a more detailed definition of Intelligent Entity:

> ...an entity is intelligent if it has an adequate model of the world (including the intellectual world of mathematics, understanding of its own goals and other mental processes), if it is clever enough to answer a wide variety of questions on the basis of this model, if it can get additional information from the external world when required, and can perform such tasks in the external world as its goals demand and its physical abilities permit.

Therefore, the goal in many fields of computer science is to make computers learn from their experience as effectively as we do. In order to achieve an intelligent behavior, we need a good representation and a good reasoning. In his proposal, John McCarthy suggests the use of deductive reasoning instead of procedural or imperative. The difference between the two types of programming is that, in declarative programming, we state what we know and what our question is. On the other hand, in procedural programming, we state the steps we want the computer to perform. The problem is how to represent the knowledge and what knowledge to represent.

1.1.2 Types of knowledge

There are two basic types of knowledge. One is the specialists's knowledge and the other is the commonsense knowledge. The specialists's knowledge is the one that mathematicians, chemical engineers and others possess. On the other side, the commonsense knowledge is the one every person has, even a small 6-year-old child. We are interested to teach the computer to reason about the world (commonsense knowledge). In addition, other important goal is to teach the computers to reason better and correctly as humans do. We really like to represent commonsense knowledge and defaults (which are universal statements). McCarthy suggested to use logic to represent knowledge. But the question is what formal language is suitable to be used. Let us review briefly the history of some languages.

1.1.3 History of languages

In the 50's the only known language was machine code. After that people started to develop languages as LISP [1] . Later, several languages have been proposed for knowledge representation. The classical logic was the main tool for representing knowledge, but it was inadequate for representation of commonsense knowledge [Baral and Gelfond, 1994]. The problem was that the classical logic is *monotonic*. A logic is *monotonic* if, for any conjunction of formulas Γ and any two formulas δ and ϕ, $\Gamma \models \phi$ implies $\Gamma \wedge \delta \models \phi$. That is: a logic is monotonic if it guarantees that adding a new fact can never make an old consequence go away [Shanahan, 2006].

However, knowledge representation is related with default reasoning, which, on the other hand, needs *non-monotonic* logic. In *non-monotonic* logic, later information may invalidate previous conclusions, without giving rise to inconsistency [Shanahan, 2006]. The most popular *non-monotonic* logics are default logic and circumscription [Baral and Gelfond, 1994]. Later, Kowalski and Colmerauer created the first logic programming language with *non-monotonic* feature called PROLOG. To handle incomplete information, Minker expanded logic programs by disjunctive information. In 1988, Gelfond and Lifschitz invented the stable model semantics for logic programming [Gelfond and Lifschitz, 1988]. Later, in [Gelfond and Lifschitz, 1991] they introduced the answer set semantics, which is the one we use in this work.

1.2 Importance of commonsense reasoning

After we have a good commonsense knowledge representation, we need to reason. There are two major reasons why we should study commonsense reasoning, as explained in

[1] LISP is a high-level programming language used for developing AI applications. Developed in 1960 by John McCarthy, its syntax and structure is very different from traditional programming languages.

[Mueller, 2006]. The first one is practical and the second one is scientific.

- **Practical**: Automated commonsense reasoning has many applications starting from intelligent user interfaces and natural language processing to robotics and vision. Commonsense reasoning can be used to make computers more human-aware, easier to use, and more flexible.
- **Scientific**: Commonsense reasoning is a core capability of intelligence that supports many other high-level capabilities. For instance, the ability to understand what is happening in a story, crucially involves commonsense reasoning. By studying commonsense reasoning we can understand better what intelligence is.

In Chapter 3 we describe with more details the properties of commonsense reasoning.

1.2.1 Example of the use of commonsense reasoning

In [Mueller, 2004], E. Mueller studies the use of commonsense reasoning to understand scripts [2] . He presents a system that understands news stories involving four terrorism scripts. In general, the steps performed by the system are the following:

1. build a commonsense reasoning problem given an information extraction template representing a terrorist incident.
2. use commonsense reasoning and a commonsense knowledge base to build a model of the terrorist incident.

where the reasoning problem, commonsense knowledge base, and model are expressed in the classical logic event calculus [Mueller, 2004]. We provide more details about the event calculus in Section 2.1.

[2] Scripts are texts involving stereotypical activities [Mueller, 2004].

The arquitecture of the system is as follows:

- The system takes as input a template about a terrorism event produced by an information extraction system used in the MUC [3] evaluations. The template is a frame with slots and slot fillers.
- The template is fed to a script classifier, which classifies what script is active in the template.
- The template and script are passed to a reasoning problem builder specific to the script, which converts the template into a commonsense reasoning problem.
- The problem and a commonsense knowledge base are passed to a commonsense reasoner, which carries out inferences and fills in missing details in order to produce a model of the input text. The model provides a deeper representation of the news story than is provided by the template alone.

The architecture of the system is shown in figure 1.1. We provide more details about this work in Chapter 6.

1.3 Definition of the problem

We want to take the *ideological conflict* domain, which involves commonsense reasoning knowledge, develop an accurate representation in ASP and solve it using Smodels. We want to create a methodology for commonsense knowledge representation in ASP and compare our approach with different methods. For example, in [Mueller, 2004], there are several stories, represented using the discrete event calculus. In general, we would like to represent commonsense knowledge in ASP and see what else has to be done in order to achieve that the computers use commonsense reasoning as we do.

[3] MUC stands for Message Understanding Conferences.

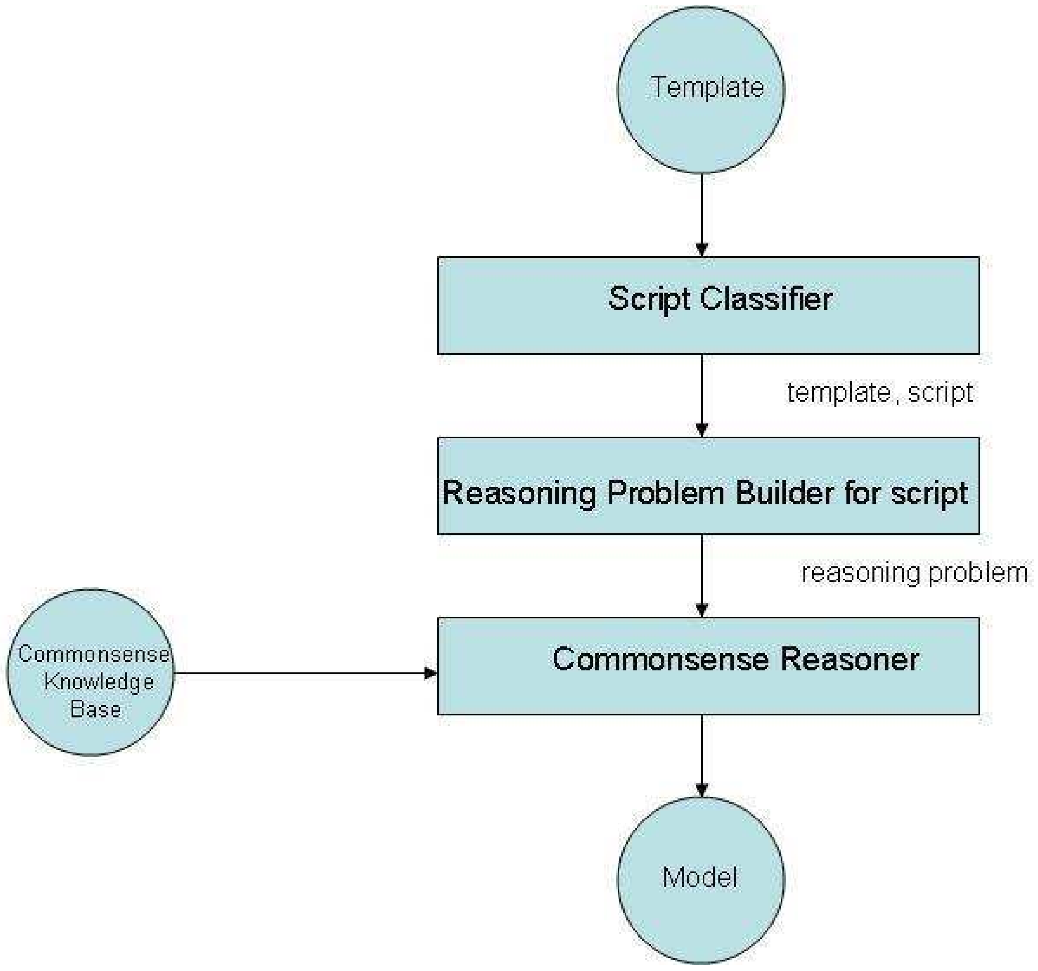

Figure 1.1: Arquitecture of Mueller's system

The goals of this work are:

- Investigate the meaning of common sense and compare the different opinions of researchers in different areas. Analyze the characteristics of commonsense knowledge and reasoning in order to be able to identify what language is ideal for representation and reasoning.
- Choose the most important definitions and propose them as a start point for our work. Given the basic definitions see what possible languages are available to cover the needs.
- Investigate the possible mathematical formalisms and decide which is the best language to use in commonsense representation and reasoning.
- Create a methodology to follow during the commonsense knowledge representation and reasoning. Identify the steps, starting from the search for a domain and ending with the knowledge representation in ASP.

1.4 Closing remarks

In this chapter, we introduced the basic notions of commonsense knowledge representation and reasoning. More details are presented in 3. We mentioned why the study of common sense is important for the field of Computer Science. We defined the problem to solve and we listed the objectives of this work. The rest of this thesis is organized as follows. In Chapter 2, we define some basic concepts and give a theoretical background. In Chapter 3, we review different points of view of what is commonsense reasoning and we conclude four basic definitions to take into account. In Chapter 4, we first propose a methodology to follow and then we represent the ideological conflict domain in ASP. In Chapter 5, we show another example of a different domain, the telephone domain, which has been already solved with event calculus. In Chapter 6, we mention the closest related work and finally, in Chapter 7 we give our results and future work.