Logical Theories for Agent Introspection

Ph.D. thesis

by

Thomas Bolander, M.Sc.

Informatics and Mathematical Modelling
Technical University of Denmark

15 September, 2003

This dissertation have been submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (Ph.D.) at the department of Informatics and Mathematical Modelling (IMM), Technical University of Denmark.

The work has been completed at IMM on a Ph.D. scholarship from the Technical University of Denmark. The work was supervised by Professor Jørgen Fischer Nilsson, IMM, and Reader Helge Elbrønd Jensen, Department of Mathematics, Technical University of Denmark.

Lyngby, 15 September, 2003

Thomas Bolander

Abstract

Thomas Bolander: Logical Theories for Introspective Agents

Artificial intelligence systems (*agents*) generally have models of the environments they inhabit which they use for representing facts, for reasoning about these facts and for planning actions. Much intelligent behaviour seems to involve an ability to model not only one's external environment but also oneself and one's own reasoning. We would therefore wish to be able to construct artificial intelligence systems having such abilities. We call these abilities *introspective*. In the attempt to construct agents with introspective abilities, a number of theoretical problems is encountered. In particular, problems related to *self-reference* make it difficult to avoid the possibility of such agents performing self-contradictory reasoning. It is the aim of this thesis to demonstrate how we can construct agents with introspective abilities, while at the same time circumventing the problems imposed by self-reference.

In the standard approach taken in artificial intelligence, the model that an agent has of its environment is represented as a set of *beliefs*. These beliefs are expressed as logical formulas within a formal, logical theory. When the logical theory is expressive enough to allow introspective reasoning, the presence of self-reference causes the theory to be prone to inconsistency. The challenge therefore becomes to construct logical theories supporting introspective reasoning while at the same time ensuring that consistency is retained. In the thesis, we meet this challenge by devising several such logical theories which we prove to be consistent. These theories are all based on first-order predicate logic.

To prove our consistency results, we develop a general mathematical framework, suitable for proving a large number of consistency results concerning logical theories involving various kinds of reflection. The principal idea of the framework is to relate self-reference and other problems involved in introspection to properties of certain kinds of graphs. These are graphs representing the semantical dependencies among the logical sentences. The framework is mainly inspired by developments within semantics for logic programming within computational logic and formal theories of truth within philosophical logic.

The thesis provides a number of examples showing how the developed theories can be used as reasoning frameworks for agents with introspective abilities.

Resumé

Thomas Bolander: Logiske teorier for introspektive agenter

Intelligente systemer (*agenter*) er generelt udstyret med en model af de omgivelser de befinder sig i. De bruger denne model til at repræsentere egenskaber ved omgivelserne, til at ræssonere omkring disse egenskaber og til at planlægge handlinger. En overvejende del af det vi sædvanligvis opfatter som intelligent handlemåde synes at involvere en evne til ikke kun at modellere ens ydre omgivelser, men også at modellere sig selv og ens egen ræssonering. Vi ønsker derfor at være i stand til at konstruere intelligente systemer som har sådanne evner. Disse evner kaldes *introspektive*. I forsøget på at konstruere agenter med introspektive evner støder man på en del problemer af teoretisk natur. I særdeleshed støder man på problemer relateret til *selvreference*, som gør det vanskeligt at sikre sig mod at sådanne agenter kan foretage selvmodsigende ræssonementer. Målet med denne afhandling er at vise hvordan vi kan konstruere agenter med introspektive evner på en sådan måde at problemerne omkring selvreference omgås.

Den model en agent har af sine omgivelser lader man sædvanligvis repræsentere ved en mængde af sætninger, der udtrykker de forestillinger som agenten har om verden. Disse sætninger formuleres indenfor en formel, logisk teori. Hvis denne logiske teori har tilstrækkelig udtrykskraft til at at tillade introspektiv ræssonering, vil tilstedeværelsen af selvreference i de fleste tilfælde forårsage at teorien bliver inkonsistent. Udfordringen kommer derfor til at bestå i at finde måder at konstruere logiske teorier på som understøtter introspektiv ræssonering, men hvor konsistens samtidig er sikret. I afhandlingen imødekommer vi denne udfordring ved at konstruere flere sådanne logiske teorier, som vi beviser at være konsistente. Disse teorier er alle baseret på første-ordens prædikatlogik.

I forbindelse med vores konsistensresultater udvikler vi et generelt matematisk værktøj, som kan benyttes til at bevise konsistensen af en lang række logiske teorier involverende forskellige former for refleksion. Den bærende idé i dette værktøj er at relatere selvreference—og andre af problemerne involveret i introspektion—til egenskaber ved bestemte typer af grafer. Dette er grafer som repræsenterer de semantiske afhængigheder imellem sætningerne i de pågældende teorier. Værktøjet er inspireret af tilsvarende værktøjer udviklet i forbindelse med semantikker for logik-programmer indenfor datalogisk logik og formelle teorier for sandhed indenfor filosofisk logik.

Afhandlingen giver et antal eksempler på hvordan de udviklede teorier kan anvendes som fundament for agenter med introspektive evner.

Preliminaries

We assume the reader to be familiar with first-order predicate logic. In addition, acquaintance with the basics of modal logic and logic programming is preferable. Experience in artificial intelligence is not required, but will be helpful. Furthermore, familiarity with basic fixed point methods such as the ones taught in introductory courses on programming language semantics or formal theories of truth will be an advantage. Some mathematical maturity will be expected, in particular in the later chapters. The difficulty level will be gradually increasing through the course of the thesis. The earlier chapters are very thorough in explaining details and discussing underlying intuition. In the later chapters, the reader is to a larger extend required to be able to grasp these things on his or her own.

Whenever a definition, lemma or theorem in the thesis is not entirely original, we will, in stating it, include a reference to the work from which it is adapted. If a definition, lemma or theorem does not contain a citation, it means that it is due to the author of this thesis. Most chapters conclude with a "Chapter Notes" section containing further bibliographic comments.

Key words and phrases: Artificial intelligence, mathematical logic, AI logics, knowledge representation, propositional attitudes, syntactical treatments of knowledge, multi-agent systems, introspection, self-reflection, self-reference, paradoxes, consistency, theories of truth, non-wellfoundedness, dependency graphs, logic programming, programming language semantics, fixed points, graph theory.

Acknowledgments

First of all, I would like to thank the following persons for showing interest in my work and for giving me many valuable comments: Roy Dyckhoff, Helge Elbrønd Jensen, João Alexandre Leite, Jørgen Fischer Nilsson, Dag Normann, Nikolaj Oldager, Stig Andur Pedersen and Graham Priest. All of these have invested considerable amounts of time and effort in reading my work and in guiding me in the right directions. I am very grateful to Jørgen Villadsen who, in the first few months of my Ph.D. studies, brought my attention to a number of important articles on which this entire thesis is based. Furthermore, I wish to thank Roy Cook, Mai Gehrke, Klaus Frovin Jørgensen, Donald Perlis and Ken Satoh for reading parts of my work, as well as for their helpful comments and encouragement.

Finally, I would like to thank my wife Audra. Even if I bought her the biggest flower bouquet in the world, it would still be very small compared to the support she gave me—and the understanding she showed me—through the writing of this thesis.

The thesis is dedicated to this sentence.

Contents

CHAPTER 1

Introduction

In this chapter we will give an informal introduction to the work presented in the thesis. We will be introducing the research area to which the present work belongs, and explain how the work contributes to the research within this area. We try to motivate the research area as a whole as well as our work within it. The chapter concludes with an overview of the structure of the thesis. Being an introduction chapter, it is aimed at a broader audience than the other chapters of the thesis.

1.1. Logic Based Artificial Intelligence

The present work falls within the research area Artificial Intelligence (AI). The main goal of artificial intelligence research is to build software and hardware systems that in some way can be conceived as behaving "intelligently". Such systems are called *agents* (note, however, that many authors use the term 'agent' in a much narrower sense). An agent can for instance be a robot placed in a real world environment to carry out certain tasks such as delivering local mail, making coffee, tidy up rooms, or even playing football (see Figure 1.1). It can also for instance be a piece of software "surfing" on the Internet to gather information on the behalf of a user. There is not a strict criterion setting artificial intelligence systems apart from other kinds of computer systems. However, one characterising property of artificial intelligence systems is that they always attempt to imitate or simulate certain aspects of human cognition. This could for instance be the aspect that humans are able to learn from experience and thereby improve performance.

This thesis belongs to the subfields of artificial intelligence known as Logic-Based Artificial Intelligence and Knowledge Representation. In these fields, one tries to build computer systems that imitate conscious-level reasoning and problem solving of humans. In this kind of reasoning, humans use sentences to express the things they know and sequences of sentences to express pieces of reasoning. Consider for instance the following simple piece of reasoning:

> There is a sign in front of me saying that vehicles higher than 2 meters can not pass through the tunnel. My truck is 3 meters high. 3 meters is more than 2 meters, so my truck will not be

FIGURE 1.1. Agents in Action (RoboCup 2002).

> able to pass through the tunnel. I should therefore look for an alternative route.

In this line of thoughts, the person (or computer?) driving the truck considers his knowledge about the world (that vehicles higher than 2 meters can not pass; that his truck is 3 meters high) and uses this as a basis for reasoning about which actions to take (to find an alternative route). The reasoning takes place in natural language, as most conscious-level reasoning performed by humans seems to do. In logic-based artificial intelligence, one seeks to imitate such language-based reasoning. For this to work, we are required to be able to equip our agents (artificial intelligence systems) with a language they can use for representing facts about the world and for reasoning about these facts. To be accessible for computations, this needs to be a formal language of some sort. Traditionally, one picks a logical language such as a language of propositional logic, a modal language or a language of predicate logic.

A first-order predicate logic formalisation of the facts used in the reasoning above could look like this:

$$\begin{aligned}
&\forall x \, (\mathit{height}\,(x) > 2 \rightarrow \neg \mathit{pass\text{-}through\text{-}tunnel}\,(x)) \\
&\mathit{height}\,(\mathit{truck}) = 3 \\
&\forall x > 0 \forall y > 0 \, (x + y > x)
\end{aligned} \tag{1.1}$$

Being strings of symbols over a fixed, finite alphabet, such logical sentences can readily be represented in a computer. But what about the reasoning that led the driver to the conclusion of not being able to pass through the tunnel? We want our artificial intelligence system to be able to derive this conclusion itself from the facts given by the logical sentences (1.1). For a computer to do this, the derivation must be purely symbolic. We actually already have a tool for such symbolic derivations: the proof theory underlying the logical language in which the sentences have been formulated. What this means is that the artificial intelligence system could derive the necessary conclusions from the represented facts by carrying out a formal proof using these facts as axioms (by means of some kind of theorem prover, for instance). Assume we take the sentences (1.1) to be axioms of a logical theory. Then, in that theory, we can carry out the following formal proof:

1.	$\mathit{height}\,(\mathit{truck}) = 3$	axiom
2.	$\forall x > 0 \forall y > 0\,(x + y > x)$	axiom
3.	$2 + 1 > 2$	instance of 2
4.	$3 = 2 + 1$	theorem of arithmetic
5.	$3 > 2$	using 3, 4
6.	$\mathit{height}\,(\mathit{truck}) > 2$	using 1, 5
7.	$\forall x\,(\mathit{height}\,(x) > 2 \rightarrow \neg \mathit{pass\text{-}through\text{-}tunnel}\,(x))$	axiom
8.	$\mathit{height}(\mathit{truck}) > 2 \rightarrow \neg \mathit{pass\text{-}through\text{-}tunnel}\,(\mathit{truck})$	instance of 7
9.	$\neg \mathit{pass\text{-}through\text{-}tunnel}\,(\mathit{truck})$	modus ponens on 6, 8

The formal proof leads to the same conclusion as the informal reasoning carried out above: the truck can not pass through the tunnel. The principal idea in logic-based artificial intelligence is thus the following: to simulate human conscious-level reasoning and problem solving by

- letting the artificial intelligence system represent facts internally as logical sentences, and
- using formal derivations from these sentences as the reasoning mechanism of the system.

1.1.1. Knowledge Bases. The set of facts represented as sentences internally in an agent is usually known as its *knowledge base*. The knowledge base contains sentences expressing those *propositions* (ideas, judgements) about the world that the agent takes to be true. We can think of these sentences as expressing the facts *known* to the agent or propositions *believed* by the agent. Consider the simple Blocks World presented in Figure 1.2. We can imagine introducing an artificial intelligence agent (a robot) into this world, given the task of moving the blocks to obtain some predefined goal configuration (e.g. building a tower consisting of all blocks piled in a specific order).

FIGURE 1.2. A Blocks World

Before the agent starts moving the blocks, it should make a plan for how to reach the given goal. This *planning* requires the agent to have knowledge about the current position of the blocks—knowledge that can be represented as sentences in the agent's knowledge base. The knowledge base could for instance contain the following sentences

$$\begin{aligned} &On\,(black, floor)\\ &On\,(dotted, black)\\ &On\,(white, floor)\,. \end{aligned} \tag{1.2}$$

The first of these sentences represents the fact that the black block is on the floor, the second that the dotted block is on the black block, etc. We will choose a slightly different way of representing these pieces of knowledge, however. To obtain a higher degree of generality, we would like to be able to represent in the agent's knowledge base not only representations of facts *known* to the agent but also for instance *beliefs* and *intentions* held by the agent (this might make the term 'knowledge base' a bit inappropriate, but we keep it for historical reasons). We therefore have to be able to distinguish between whether $On\,(black, floor)$ expresses a fact *known* to the agent or for instance simply expresses a state of affairs *intended* by the agent. To indicate that the sentences (1.2) express knowledge, we instead write

$$\begin{aligned} &K\,On\,(black, floor)\\ &K\,On\,(dotted, black)\\ &K\,On\,(white, floor)\,, \end{aligned}$$

where $K \ldots$ stands for "it is known that ...". When we want to express belief rather than knowledge, we write $B \ldots$ to mean that "it is believed that ...".

The situation is presented in Figure 1.3, where the thought balloon is used to represent the knowledge base of the agent. Notice the simple relationship existing between the objects in the knowledge base and the objects in the agent's environment: the objects in the knowledge base are sentences, each of which expresses the relative position of exactly two of the objects in the environment. The knowledge base of the agent is a *model* of the agent's

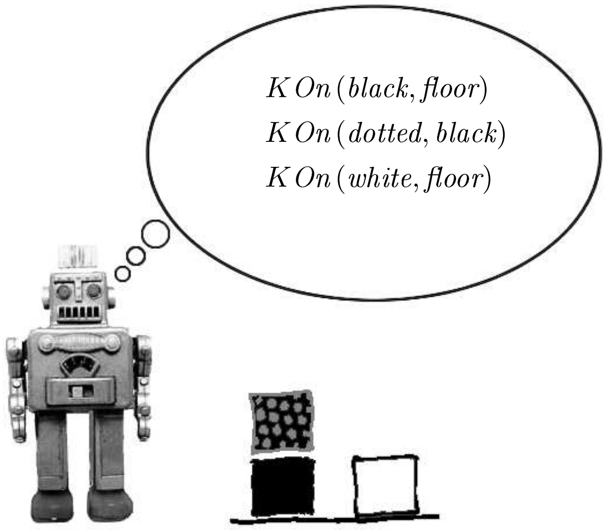

FIGURE 1.3. An Agent in the Blocks World

environment, since the objects in the knowledge base represent (or *model*) properties of the objects in this environment. It is the agent's ability to model its own environment that makes it able to reason about this environment and to predict the consequences of performing various actions. It is thus its ability to model its own environment that makes it able to *plan* its actions. This is also the case with humans: when we reason about which actions to take in a given situation (for instance when planning our next move in a game of chess), we try to predict the consequences of each possible action by using our knowledge about the world, that is, our *model* of the world, and then pick the action with the most preferred consequences. We can think of our brains as containing a "map" of the world—the world as we conceive it—and the knowledge base of an agent is playing a similar role to the agent as this "map" does to us.

1.2. Introducing Introspection

We have now introduced the basic idea underlying the field of logic-based artificial intelligence, and explained that it is based on trying to imitate certain aspects of human cognition. This thesis concerns the possibility of implementing yet another aspect of human cognition in artificial intelligence systems: *introspection*. It is the aim of the thesis to contribute to the theoretical foundations of constructing agents with the ability to introspect. But what is introspection? The Oxford English Dictionary defines it as "examination or observation of one's own thoughts, feelings or mental state". The MIT Encyclopedia of Cognitive Science defines it as "the process by which people come to be attentively conscious of mental states they are currently in". Correspondingly, to have introspection in an artificial intelligence system means that the system is able to reflect on its own knowledge (or ignorance), its own reasoning, actions, and planning. The father of artificial intelligence, John McCarthy, puts it this way: "We say that a machine *introspects* when it comes to have beliefs about its own mental state" [McCarthy, 1979]. There are various degrees to which one can have introspection. Some simple examples of introspective beliefs are the following:

- I do not know how to get from here to Skørping.
- I believe that some of my beliefs are false.
- I have no knowledge about a striped block.
- I believe that Alice believes that I have the ace of hearts.
- I believe that I have insufficient knowledge about this problem to be able to solve it.
- I believe that Alice knows more about the problem than I do.
- Every time I start believing something, it always turns out to be false.
- I do not believe this very sentence.

The reason that one wants to equip artificial intelligence agents with the ability to introspect is that "much intelligent behaviour seems to involve an ability to model one's environment including oneself and one's reasoning" [Perlis and Subrahmanian, 1994]. We know that self-reflection plays a central role in human cognition—it is one of the primary abilities setting us apart from animals—and we would therefore expect this ability to play an equally important role in artificial intelligence. We use introspection whenever we reason about the way we carry out certain tasks, and whenever we reason about how to improve our routines for carrying out these tasks. Thus, introspection is fundamental for our ability to consciously improve ourselves. Introspection is also needed when we want to reason about the things that we do not know, but that we might need to acquire knowledge of to carry out certain tasks (an example of this in an AI context will be given in the following section). Furthermore, whenever two or more persons co-operate in reaching a goal (or are opponents competing to reach opposite goals), they will reason about each others statements, beliefs, actions, and plans, and thus they will indirectly be reasoning about themselves (through each other). This is particularly evident in game-playing situations: If Alice and Bob are players in a two-person game, then Alice will—in producing her strategy—try to predict the strategy of Bob, and his strategy will, in turn, involve trying to predict the strategy of Alice. Thus both Alice and Bob will indirectly introspect through each other. This makes introspection particularly important in connection with *multi-agent systems*: systems consisting of several independently functioning agents acting, communicating and co-operating in a common environment. An example of a multi-agent system is the football playing environment presented in Figure 1.1, p. 14.

The arguments we have now given all show introspection to be an important cognitive ability. Indeed, as McCarthy puts it, "consciousness of self, i.e. introspection, is essential for human level intelligence and not a mere epiphenomenon" [McCarthy, 1996]. These are some of the main reasons we have for wanting to build artificial intelligence systems with introspective abilities. Further arguments for the necessity of introspection in artificial intelligence can be found in, among others, Perlis [1985; 1988], Perlis and Subrahmanian [1994], Konolige [1988], Fasli [2003], Kerber [1998] and Grant *et al.* [2000]. In this thesis we will show how to equip artificial intelligence agents with a language expressive enough to allow a high degree of introspection. We will then give some examples of how these increased powers can be used by the agents to improve their performance.

1.3. Formal Introspection

1.3.1. An Example. Let us give a simple example of a situation in which an artificial intelligence agent will need a basic form of introspection to carry out the reasoning we expect it to do. We consider again the situation presented in Figure 1.3. Now assume that we add a fourth block to the world, but keep everything else the same, hereby obtaining the situation presented in Figure 1.4. Suppose a 'human user' asks the agent (robot) to put the striped block on top of the dotted one. The problem is that the agent's knowledge base is left unchanged, so the agent does not know anything about the striped block. The right thing for the agent to do in this case would be to ask the user about the position of the striped block, but in order to do this the agent must *register the fact that it does not know where the new block is.* This requires introspection, since the agent must introspectively look into its own knowledge base and from this arrive at the conclusion that no information about a striped block is present in the base. To express such reasoning, the agent needs a new set of sentences. The point is that in order for the agent to say "I do not know the position of the striped block" we have to allow the agent to refer to its own beliefs as objects in its world. In order for the agent to express facts concerning its own knowledge, we have to iterate on the occurrences of K. To express that the agent *knows* that it does *not know* that the striped block is

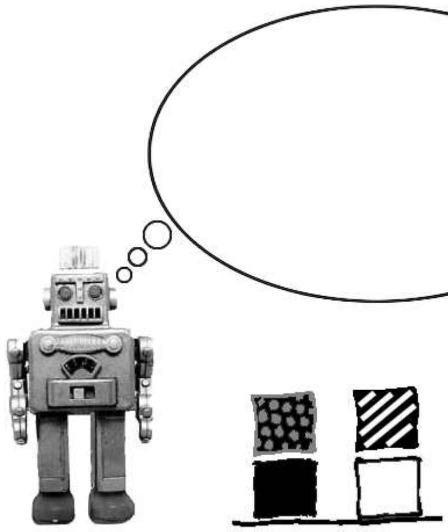

FIGURE 1.4. Adding a New Block to the Blocks World

on the white block, we can write

$$K\neg K\, On\,(striped, white)\,.$$

This sentence expresses an agent's introspective awareness of the fact that it does not know the striped block to be on the white block. It corresponds to the agent saying: "I do not know that the striped block is on the white block" (assuming, for now, that everything said by the agent is known to the agent). Using iterated occurrences of K the agent can also express things such as:

- "I do not know whether the striped block is on the floor or not":

$$K\,(\neg K\, On\,(striped, floor) \wedge \neg K \neg\, On\,(striped, floor))\,.$$

- "I do not have any knowledge about the position of the striped block":

$$K\forall x\big(\neg K\, On\,(x, striped) \wedge \neg K\neg\, On\,(x, striped) \wedge \\ \neg K\, On\,(striped, x) \wedge \neg K \neg\, On\,(striped, x)\big). \quad (1.3)$$

The sentences above are both cases of *meta-knowledge*: knowledge about knowledge. Using meta-knowledge in reasoning is the most basic kind of introspection. Sentence (1.3) expresses the introspective insight needed by the agent to realise that it has to acquire further knowledge to solve the task given by the user (the task of putting the striped block on top of the dotted one). The use of introspection in reasoning about what you do not know is treated in depth in, among others, Konolige [1988] and Kraus *et al.* [1991].

1.3.2. The Syntactic Status of K. We have not yet said anything about the syntactic status of the K's (and the B's to express belief). What is the syntactic relationship between a sentence φ and the sentence $K\varphi$ expressing that φ is known? We will at this point only touch on the issue quite briefly to keep the introduction as clean from technical involvement as possible. The problem will be discussed more thoroughly in Section 2.4. There are essentially two different ways to treat K: either it is a *modal operator* (a propositional operator, a one-place connective) or it is a *predicate*. If K is a predicate and φ is a sentence, then $K\varphi$ it is actually an abbreviation for $K(\ulcorner\varphi\urcorner)$, where $\ulcorner\varphi\urcorner$ is a term denoting φ (a *name* for φ). In terms of expressiveness, the predicate treatment is much stronger than the operator treatment (at least in the context of first-order formalisms). The reason is that if x is a variable then Kx is a well-formed formula in the predicate treatment but not in the operator treatment: x is a term and not a well-formed formula, and operators only apply to formulas, whereas predicates only apply to terms. This means that only in the predicate treatment will it be possible for the knowledge base of the agent to contain strongly introspective knowledge such as:

- "The agent knows that some of its beliefs are false":
$$K\exists x\,(Bx \wedge \neg\, True(x))\,.$$
- "The agent knows that Alice knows at least as much about skiing as itself":
$$K\forall x\,(About\,(x, skiing) \wedge Kx \to K_{Alice}\, x)\,.$$
- "The agent knows that is has no knowledge about a striped block":
$$K\neg\exists x\,(About\,(x, striped) \wedge Kx)\,. \tag{1.4}$$
- "The agent knows that in order to move an object it must have some knowledge about the object":
$$K\forall x\,(\neg\exists y\,(About\,(y, x) \wedge Ky) \to \neg\, Can\text{-}move\,(x))\,.$$

These sentences all express introspective knowledge that should be accessible to any agent we claim to be strongly introspective. Since the sentences can only be expressed within the predicate approach (they all contain either K or B applied directly to a variable), we will choose this approach throughout the thesis. Note that the sentence (1.4) gives a simpler and more general formalisation of the agent's ignorance concerning the striped block than the sentence (1.3) considered above (assuming that we have given a suitable formalisation of the *About* predicate). Only the less general sentence (1.3) could be part of an agent's knowledge base on the operator approach. The operator approach only gives a rather limited kind of introspection, since in this approach all we have is that whenever φ is a sentence in our language, there is another sentence

$K\varphi$ expressing "φ is known" (or "φ is in the knowledge base"). The operator approach does not allow us to express statements such as "the agent knows that it does not have any contradictory beliefs": $K\neg\exists x\,(Bx \wedge B\,not(x))$.[1]

1.4. The Problem with Introspection

We have now decided which language to equip our artificial intelligence agents with: first-order predicate logic with knowledge and belief treated as predicates. We have argued that this language is expressive enough to allow the agents to express strongly introspective beliefs. It seems that there is then only a short step to actually build these agents using either a logic programming inference engine or a general first-order theorem prover. Unfortunately, the step is not as short as it seems. The reflective character of strong introspection makes it vulnerable to self-referential reasoning which can cause *paradox* and *inconsistency* in the knowledge base.

A *paradox* is a "seemingly sound piece of reasoning based on seemingly true assumptions, that leads to a contradiction (or other obviously false conclusion)" (The Cambridge Dictionary of Philosophy). If an agent engages in a paradoxical piece of reasoning, it will introduce a contradiction into its knowledge base. In the worst case this will make the agent useless, since from a contradiction the agent will be able to derive any conclusion, whether true or false (if the logic underlying the agent's reasoning is classical). Thus we should prevent our agents from being able to engage in paradoxical reasoning. But with strong introspection and its close relative *self-reference*, this is not an easy matter.

1.4.1. The Knower Paradox. It is well-known that self-reference can cause paradox when used in connection with the concept of truth. A classical example of this is the *liar paradox* The liar paradox is the contradiction that emerges from trying to determine whether the *liar sentence* is true or false. The liar sentence is the self-referential sentence L saying of itself that it is not true. We can express L in the following way

$$L : \text{sentence } L \text{ is not true.}$$

The reasoning that leads to a contradiction goes as follows:

> If the sentence L is true, what it states must be the case. But it states that L is not true. Thus if L is true, it is not true. On the contrary assumption, if L is not true, then what it states must not be the case and, thus, L is true. Therefore, the liar sentence L is true if and only if it is not true. This is a contradiction.

[1] We assume here that the function symbol *not* represents a function mapping names of formulas to the names of their negations.

More recently, it has been shown that self-reference can also cause paradox when used in connection with the concepts of knowledge and belief [Kaplan and Montague, 1960; Montague, 1963; Thomason, 1980]. This turns out to be a serious threat to the ambition of building strongly introspective artificial intelligence systems. The main goal of the thesis is to show how these problems can be circumvented without sacrificing neither the underlying classical (two-valued) logic, nor the ability to form strongly introspective beliefs.

The simplest paradox of self-reference and knowledge is the *knower paradox* (or the *paradox of the knower*) [Kaplan and Montague, 1960]. We will give a version of it here. The knower paradox is closely related to the liar paradox. It is based on the *knower sentence* S given by

$$S : \text{sentence } S \text{ is not known by Mr. X.}$$

This sentence is obviously *self-referential*: it expresses a property concerning the sentence itself (the property of not being known by Mr. X). Reasoning about whether S is known by Mr. X or not turns out to lead to a paradox. Let us first prove the fact that

$$\text{Sentence } S \text{ is not known by Mr. X.} \tag{1.5}$$

The argument leading to this conclusion goes as follows:

> Assume to obtain a contradiction that S is known by Mr. X. Then S must be true (nothing false can be *known*, only *believed*). Thus, what S states must be the case. But it states that it itself is not known by Mr. X. In order words, S is not known by Mr. X. This contradicts the assumption that S is known by Mr. X, so that assumption must be false.

The argument shows that S is *unknowable* to Mr. X: Mr. X can never correctly know S to be true, since assuming this leads to a contradiction. Note that since our conclusion (1.5) is actually nothing more than the sentence S itself, we have also proved that S is indeed true! Thus we have the counter-intuitive consequence that there exists a *true* sentence (the sentence S) which Mr. X can *express* but which he can never *know*.[2] It seems to prove that that Mr. X will always be under some kind of restriction regarding what he can know. Interestingly, S can easily be known by *another* agent Mr. Y—only can it not be known by Mr. X, since it is a proposition concerning Mr. X himself. The situation has a much more disturbing consequence than this, however. The piece of reasoning given above—leading to the conclusion (1.5)—should be accessible to anyone with sufficient reasoning capabilities. In particular,

[2]It is not a coincidence that this conclusion appears to be closely related to Gödel's First Incompleteness Theorem [Gödel, 1931] saying approximately: There exists a *true* sentence which formal arithmetic can *express* but which formal arithmetic can not *prove*.

Mr. X should be able to carry out the argument himself, if he possesses the sufficient reasoning powers. The result of Mr. X carrying out this piece of reasoning will be that he himself comes to know that (1.5) holds. Thus we get

Mr. X knows that (1.5) holds.

But since (1.5) is simply the sentence S itself, we can replace (1.5) by S and get

$$\text{Mr. X knows that } S \text{ holds.} \tag{1.6}$$

What we have now is a *paradox*: Our two conclusions (1.5) and (1.6) are in immediate contradiction with each other. This is the *knower paradox*.

1.4.2. Formalising the Knower Paradox. The sentence S on which the knower paradox is based is certainly of a somewhat pathological nature, since it is merely expressing a claim concerning its own epistemological status (as the liar paradox is merely a sentence expressing a claim concerning its own semantic status). The fact that S is pathological could reasonably lead us to the conclusion that we should not worry about the sentence and any counter-intuitive consequences it might have. Unfortunately, it is a sentence that can easily be formalised within the logical framework for introspective agents we have been presenting. Let us show how. Let φ be the following sentence

$$\forall x \left(Dx \to \neg Kx \right).$$

Suppose the predicate Dx is chosen or constructed such that it is satisfied exactly when x is φ (or, more precisely, when x is the *term* denoting φ). There does not seem to be any reasons why an agent should not have access to a predicate such as Dx: it simply picks out a particular sentence in the language of the agent. The sentence φ expresses that the sentence picked out by the predicate Dx is not known by the agent. But the sentence picked out by Dx is the sentence φ itself. Thus the sentence φ expresses that φ is not known by the agent. It is therefore *self-referential* in the same way as the liar sentence L and the knower sentence S considered above. If we name the agent that φ concerns 'Mr. X', then the proposition expressed by φ is the exact same as the proposition expressed by the knower sentence S. Actually, we can now within our logical framework recapture the entire informal reasoning carried out above—using the sentence φ as the formal counterpart to the knower sentence S. First of all, we can prove that

$$\text{The sentence } \neg K\varphi \text{ is true.} \tag{1.7}$$

This conclusion is the formal counterpart to (1.5) given above. The argument leading to (1.7) is the following:

> Assume to obtain a contradiction that $\neg K\varphi$ is false. Then $K\varphi$ is true. Thus the agent knows φ, and so φ must be true. Therefore, $\forall x\,(Dx \to \neg Kx)$ is true, and since $D\varphi$ is true, $\neg K\varphi$ must be true as well. But this contradicts the assumption that $\neg K\varphi$ is false, so that assumption must be false.

Now this piece of reasoning—leading to the conclusion (1.7)—should be accessible to any agent with sufficient reasoning capabilities. In particular, the agent with which the argument is concerned should be able to carry it out itself. The result of the agent carrying out this piece of reasoning will be that it comes to know that $\neg K\varphi$ is true, that is, we get that $K\neg K\varphi$ is true. But now consider the following argument:

> We are given that the sentence $K\neg K\varphi$ is true. Since φ is the only object satisfying Dx, we have that $\neg K\varphi$ is equivalent to $\forall x\,(Dx \to \neg Kx)$. Thus we can replace $\neg K\varphi$ by $\forall x\,(Dx \to \neg Kx)$ in $K\neg K\varphi$, which gives us the sentence
>
> $$K\forall x\,(Dx \to \neg Kx)\,.$$
>
> But this is the same as $K\varphi$. We have thus shown that $K\neg K\varphi$ is equivalent to $K\varphi$, and since $K\neg K\varphi$ is true, this proves $K\varphi$ to be true.

This argument shows that we have the following counterpart to (1.6):

$$\text{The sentence } K\varphi \text{ is true.} \tag{1.8}$$

Now again we have a contradiction: (1.7) and (1.8) are in immediate contradiction with each other. What we have obtained is a *formalisation* of the knower paradox (a more detailed presentation of the formalities of this paradox will be given in the proof of Theorem 4.5 in Section 4.2.2). Since the contradiction is now derived completely within our logical framework, it shows our reasoning framework for introspective agents to be inconsistent! Thus there must be a serious flaw in the proposed framework.

The knower paradox shows that mixing self-reference with the concept of knowledge can form an explosive cocktail. Thus we have to be very careful whenever we deal with situations in which we have both of these ingredients. One such situation occurs when we try to implement introspection in knowledge-based agents, since along with introspection we automatically get self-reference. In particular, we can within the logical framework for these agents construct a self-referential sentence φ as above (a *knower sentence*). Reasoning about this sentence leads to a contradiction, showing the framework to be inconsistent. To obtain a useful framework for introspective agents, we must therefore somehow avoid self-reference or limit its harmful influences.

Most of the thesis will concern different strategies to avoid self-reference or to make it innocuous in connection with introspective agents.

1.5. Avoiding the Problems of Introspection

We have been showing that equipping agents with an ability to form strongly introspective beliefs allows these agents to form self-referential statements, and that reasoning about these statements can lead to paradox. The problem we are facing is therefore to tame self-reference in a way that allows us to avoid paradox but still keep the expressiveness and strong reasoning powers of introspection. To do this, we must take a closer look at how self-reference enters the picture, and what can be done to avoid the problems that it carries along.

1.5.1. Modelling Oneself. Let us first try to explain why self-reference enters the picture, and why it is not a trivial matter to avoid it. Consider again the agent in Figure 1.3. This agent is non-introspective, since none of the sentences in the knowledge base express propositions concerning the agent itself. As previously mentioned, we can think of the knowledge base of an agent as a *model* that this agent has of its world. Any non-introspective agent only models its external environment, that is, all items in its knowledge base concern the external world only. This means that we have a complete separation between the model (the knowledge base) and the reality being modelled (the external environment). The situation is much more complicated and problematic with introspective agents. These differ from non-introspective ones by modelling not only their external environment but also themselves. It is by having models of themselves they are given the ability to introspect (humans also have models of themselves, since we generally *believe* that we can predict our own reactions to most situations that can occur to us. We rely heavily on this ability when we plan our actions). Modelling not only one's environment but also oneself means that we can no longer have a complete separation between the model (the knowledge base) and the reality being modelled (the world including the agent and its knowledge base). It is as a consequence of this lack of separation between *that which models* and *that which is being modelled* that we are able to form sentences concerning themselves, that is, self-referential sentences. Self-reference is made possible by the fact that introspection breaks down the separation between model and modelled reality. Thus self-reference is automatically introduced along with strong introspection, and this makes it very difficult to get rid of self-reference (or its harmful effects) while still keeping the introspection.

The problematic thing about self-reference is that it allows us to form sentences that express properties only concerning themselves. These are sentences of a somewhat pathological nature, since they do not model an external reality, but only themselves. Some of these sentences express properties of themselves that are opposite of the properties that they actually possess. This is precisely what makes them paradoxical, and what threatens to contaminate the entire knowledge base with paradox (as is the case with the knower paradox). It is important to note that although these sentences are indeed pathological, they are still natural consequences of modelling a part of reality containing the model itself.

1.5.2. Dependency Graphs. To be able to avoid the problems imposed by self-reference, we must first of all be able to identify situations in which self-reference occur and separate these from situations in which no self-reference is involved. This requires us to have a way of mathematically representing *relations of reference* through which we can give self-reference a precise meaning. In the thesis, we show how to do this using graphs. Using these graphs and their properties, we are able to prove a number of general results relating questions of the presence of self-reference (and related phenomena) to questions of consistency and paradox-freeness. From these results we are then able to construct a number of logical theories allowing strong introspection but avoiding the problems of self-reference. In particular, we are able to construct logical theories that circumvent the knower paradox.

We now give a brief sketch of our graph theoretical approach to reference and self-reference. The idea is to represent relations of reference by a directed graph. We take the nodes of this graph to be the sentences of the language that our agents use. In the graph we then put an edge from a node p to a node q whenever p is a sentence *referring* to the sentence q. If p is the sentence $K\,On(b,d)$ and q is the sentence $On(b,d)$, then we will have an edge from p to q since the sentence $K\,On(b,d)$ ("the agent knows that the <u>b</u>lack block is on the <u>d</u>otted block") refers to the sentence $On(b,d)$ ("the black block is on the dotted block"). Similarly, the sentence $K\neg K\,On(b,d)$ refers to $\neg K\,On(b,d)$, so we would have an edge from $K\neg K\,On(b,d)$ to $\neg K\,On(b,d)$ as well. Furthermore, we choose to have an edge from a sentence φ to a sentence ψ whenever ψ is an immediate constituent of φ (for instance when φ is the sentence $\neg K\,On(b,d)$ and ψ is the sentence $K\,On(b,d)$). Finally, we have an edge from φ to ψ whenever ψ is a substitution instance of φ (for instance when φ is $\forall x\,(Dx \to \neg Kx)$ and ψ is $D\,On(b,d) \to \neg K\,On(b,d)$).

The obtained graph is called a *dependency graph*. Since we are thinking of it as representing relations of *reference*, it may seem more appropriate to call it a *reference graph*. However, our choice of name relies on the fact that our concept is closely related to other notions of dependency graphs used in various fields within computer science (in particular, our graphs are closely related to the dependency graphs used in logic programming [Apt *et al.*, 1988]). We say that a sentence φ *refers indirectly* to a sentence ψ if there is a path from φ to ψ in the dependency graph. The following is an example of a path in a dependency graph

$$K\neg K\,On\,(b,d) \longrightarrow \neg K\,On\,(b,d) \longrightarrow K\,On\,(b,d) \longrightarrow On\,(b,d)\ .$$

This path shows that the sentence $K\neg K\,On\,(b,d)$ refers indirectly to the sentence $On\,(b,d)$. This is in accordance with our standard use of the notion 'indirect reference': when we say that it is known that it is not known that the black block is on the dotted block, then we make an indirect reference to the relative position of the two blocks expressed by the sentence $On\,(b,d)$. We now say that a sentence φ is *self-referential* if it refers indirectly to itself, that is, if φ is contained in a *cycle* in the dependency graph. Thus cases of self-reference in the language become identified as cycles in the associated dependency graph. In Section 1.4.2 we claimed the sentence φ given by $\forall x\,(Dx \to \neg Kx)$ to be self-referential. This is consistent with the definition of self-referentiality just given. To see this, consider the following subgraph of the dependency graph:

$$\begin{tikzcd}
 & \varphi \overset{df}{=} \forall x\,(Dx \to \neg Kx) \arrow[dr] \arrow[drr] & & \\
K\varphi \arrow[ur] & \circlearrowright & D\varphi \to \neg K\varphi \arrow[dl] \arrow[dr] & \cdots \\
 & \neg K\varphi \arrow[ul] & & D\varphi
\end{tikzcd}$$

It is seen that φ is contained in a cycle, and thus it is self-referential by the definition given.

Using dependency graphs, we can now always check whether a particular sentence is self-referential or not. By considering such graphs we can see which sentences we need to remove from our language to avoid self-reference and thus avoid paradoxes. In some cases we want to keep all the self-referential sentences, but then we can use the graphs to see how the underlying logic can be weakened so that these sentences will do no harm. In both cases, what we obtain are consistent formalisms that can be used as the reasoning frameworks for introspective agents.

1.6. Summing Up

Let us briefly sum up the content of this chapter. We have argued that introspection is a crucial cognitive ability that we would like artificial intelligence agents to have. We have then shown that first-order predicate logic with knowledge and belief treated as predicates is an expressive formalism in which strongly introspective representation and reasoning can take place. We therefore propose to use this formalism as the framework for building introspective agents. Unfortunately, it turns out that a bi-product of the high expressiveness of the formalism is that agents using it will be able to engage in paradoxical reasoning, and thus become useless. The possibility of engaging in paradoxical reasoning is caused by the presence of self-referential sentences in the language. Self-reference is automatically introduced together with introspection, since introspection implies that one can no longer separate one's model of the world from this world itself. To save the formalism from paradox, we need to either exclude the self-referential sentences from the language or ensure that all self-reference becomes innocuous. By introducing dependency graphs, we are able to identify self-referential sentences, and through these graphs we are then able to give paradox-free (consistent) formalisms that still have the expressiveness needed in strong introspection.

1.7. Short Overview of the Thesis

The thesis is organised as follows. In Chapter 2 we will describe different views and intuitions towards the formalisation of knowledge within a logical theory, and then explain the views and intuitions underlying this thesis. We will be discussing the advantages of formalising knowledge within first-order predicate logic. In Chapter 3 we give a detailed definition of the languages and theories of predicate logic to be used in the thesis. In Chapter 4 we present the inconsistency problems related to the formalisation of introspective knowledge. We also present a number of approaches to circumvent these inconsistency problems. In Chapter 5 we present an original approach to the problem. This approach involves using graphs and graph-based methods. We apply these methods in Chapter 6 to strengthen a number of the previously known consistency results. These strengthened results are Theorem 6.9 and Theorem 6.23, which are the main results of the thesis. The chapter includes a number of examples of how the results can be applied in constructing agents that can safely do introspective reasoning. The thesis ends with a conclusion.

CHAPTER 2

Setting the Stage

As mentioned in the previous chapter, our overall goal is to construct logical formalisms to serve as the reasoning mechanisms for introspective agents. There are many choices that have to be made in this connection: choices of underlying logical framework; choices of how statements within the logical framework are intended to be interpreted; choices of which aspects of the problem to include and which to leave out, etc. In the following we will describe and motivate the choices we have made in relation to the work of this thesis. We will describe different possible views and intuitions towards the formalisation of knowledge and belief within a logical theory, and explain the views and intuitions underlying this thesis. The detailed mathematical definitions of the logical theories to be used are left for the next chapter.

2.1. Knowledge Bases as Logical Theories

The 'consciousness' of an artificial intelligence agent is its *knowledge base* in which its knowledge about the world is recorded. Known facts are represented within the knowledge base as sentences of some formal language. Thus the knowledge base can be thought of as simply being a set of formal sentences. However, the agent is required to be able to reason about the world using the facts about the world represented in its knowledge base. Therefore the knowledge base needs to be equipped with some machinery for manipulating the sentences in the base and for deriving new consequences of the facts represented by these sentences.

Suppose, for instance, that the knowledge base of an agent consists of the following sentences

$$\begin{aligned} &K\, On\,(dotted, black) \\ &K \forall x \forall y\,(On\,(x,y) \rightarrow \neg On\,(y,x))\,. \end{aligned} \tag{2.1}$$

This situation is illustrated in Figure 2.1. The first sentence expresses the agent's knowledge of the fact that the dotted block is on the black block, and the second sentence expresses the agent's knowledge of the fact that the *On*-relation is asymmetric. If the agent is asked whether the black block is on the dotted block, it can not answer this question by a simple look-up in

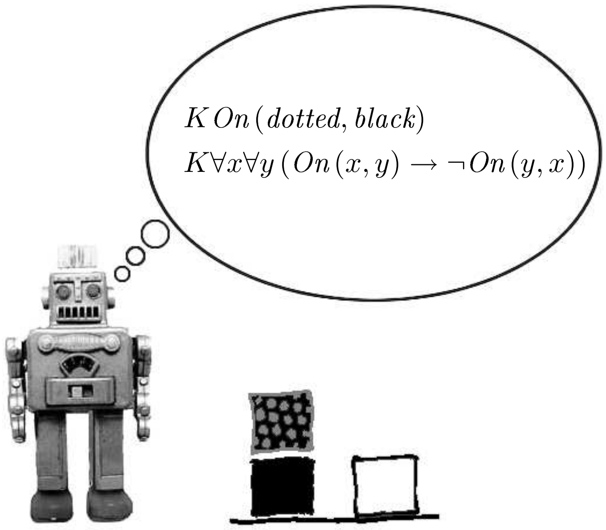

FIGURE 2.1. Another Agent in the Blocks World

its knowledge base, since the knowledge base does neither explicitly contain a proposition claiming that the black block is known to be on the dotted block:

$$K\,On\,(black, dotted)\,,$$

nor does it contain a proposition claiming that the black block is known *not* to be on the dotted block:

$$K\neg On\,(black, dotted)\,.$$

Nevertheless, the fact that the black block is not on the dotted block is implicitly represented in the knowledge base, since it is a consequence that can be logically deduced from the fact that the dotted block is on the black block and the fact that the *On*-relation is asymmetric. We would certainly want the agent to be able to deduce this consequence from the facts represented in its knowledge base. The simplest way to obtain this is to provide the agent with logical inference rules that it can use to derive new facts from the facts already given.

This leads to the idea of taking the agent's knowledge base to be a *logical theory* rather than simply a set of formal sentences. In a logical theory we have *axioms* to represent known facts and *inference rules* to allow the agent to derive new facts, *theorems*, from the known facts. If we take the knowledge base of an agent to be a logical theory, then the reasoning mechanism of the agent is nothing more than a mechanism for making formal proofs within the logical theory. This means that the reasoning mechanism can be implemented in a computer simply by using the inference engine of a logic programming interpreter or a suitable theorem prover.

2.2. Explicit Versus Implicit Knowledge

If the knowledge base of an agent is a logical theory, how should we then think about the axioms, proofs and theorems of this theory? First of all, the knowledge base is not necessarily a static theory. At every point in time, the knowledge base is a logical theory, but this logical theory may evolve over time as new observations are made and new facts are learned. If the reasoning mechanism of an agent is a theorem prover proving theorems of some logical theory, then it seems sensible that whenever a new theorem is proved (a new fact is learned), it is added as an axiom to the theory. This implies that the logical theory will expand over time.

Suppose the knowledge base of an agent at some point in time is given as a logical theory S of some suitable formal language. If $K\varphi$ is an axiom in S, we say that φ is *explicitly known* by the agent. If $K\varphi$ is a theorem in S but not an axiom, we say that $K\varphi$ is *implicitly known* by the agent. If S is the theory consisting of the two axioms (2.1), then the agent explicitly knows that the dotted block is on the black block. It does not explicitly know that the black block is *not* on the dotted block, but it might implicitly know this if S contains the sufficient inference rules for $K\neg On(black, dotted)$ to be a theorem (see Section 3.7 below).

We will not be too concerned with the distinction between explicit and implicit knowledge in this thesis. Since our primary focus is to find ways to ensure that certain things can never be inferred by our agents (such as paradoxes), we focus primarily on implicit knowledge. When we talk about knowledge we will therefore most often be referring to implicit knowledge. If we say that an agent knows φ, what we mean is that $K\varphi$ is a theorem of the logical theory that constitutes the agent's knowledge base.

2.3. External Versus Internal View

As mentioned in Chapter 1, we can think of the knowledge base of an agent as the *model* that this agent has of its environment. We have chosen to represent this model as a logical theory. The agent uses the logical theory to reason about its environment. But such a logical theory also has another possible use. It could just as well be a theory that *we* humans use to reason *about* the agent. That is, the logical theory could be considered to be *our* model of the agent rather than the *agent's* model of its world. Then *we* can use the theory to prove various properties of the agent. When we think of the logical theory in this way, the agent itself does not have to be of a type involving any kind of explicitly represented knowledge. The agent could for instance be a very simple reactive system such as a thermostat. In this case, when we use the logical theory to prove that the system or agent *knows* certain facts, this is simply knowledge that we *ascribe* to the agent—not necessarily knowledge that it explicitly possesses. We can ascribe a thermostat the knowledge that if it gets to cold the heat must be turned on, but this is not a piece of knowledge explicitly represented within the thermostat [McCarthy, 1979].

Consider again the theory consisting of the two axioms (2.1). We may use this theory to *describe* any agent that can correctly answer questions about the relative position of the black and the dotted block. We can do this independently of the kind of internal representation of facts (if any) that the agent actually has. When we for instance prove that $K\neg On(black, dotted)$ holds in the theory, this means that we *ascribe* to the agent the knowledge that the black block is not on the dotted block. There are many different ways in which this piece of knowledge could be represented within the agent itself, and it might only be represented very indirectly.

Thus the logical theories become *our* tool for reasoning about the behaviour of agents and for proving facts about them. Such reasoning can be very useful in analysing, designing and verifying various types of computer systems—in particular distributed computer systems, where each automaton (process) within the system is modelled as an individual agent having its own knowledge and belief [Fagin *et al.*, 1995].

The view of a logical theory as a tool for us to reason about agents is called the *external view* (or the *descriptive view*). The view of a logical theory as the reasoning mechanism within an agent itself is called the *internal view* (or the *prescriptive view*). In this thesis we are mostly concerned with the internal view, although most of the results we obtain apply equally well to the external view. In fact, we will not be explicitly distinguishing between the two views. If a logical theory S is describing the knowledge base of an agent and if $K\neg On(black, dotted)$ is provable within that theory, then we simply say that the agent knows $\neg On(black, dotted)$. Whether this is only knowledge we *ascribe* to the agent, or we think of it as knowledge that the agent has itself inferred using the axioms and inference rules of S, is not of central importance to the things we have to say.

2.4. Operator Versus Predicate Approach

It is common to distinguish two approaches to the question of how to syntactically represent knowledge and belief in a logical theory [Whitsey, 2003;

des Rivières and Levesque, 1988]. These two approaches are called the *operator approach* and the *predicate approach*. The traditional as well as currently dominating approach is the *operator approach* which goes at least back to Hintikka [1962]. In this approach, K and B are *sentential operators* (*modal operators*) that are applied directly to formulas. The operator approach is usually combined with a *possible-world semantics* [Kripke, 1963]. Often the operator treatment of knowledge and belief is realised within a propositional modal logic, but various first-order modal logics have been proposed as well [Levesque, 1984; Lakemeyer, 1992; Levesque and Lakemeyer, 2000] (general presentations of first-order modal logic can e.g. be found in Hughes and Cresswell [1996] or Fitting and Mendelsohn [1998]).

The second approach is the *predicate approach*. In this approach, K and B are *predicates* of a first-order predicate logic, and they are applied to *names* of formulas (which are terms) rather than the formulas themselves. There have been given many arguments in favour of the predicate approach over the traditional operator approach [Asher and Kamp, 1986; Attardi and Simi, 1995; Carlucci Aiello *et al.*, 1995; Davies, 1990; Fasli, 2003; Konolige, 1982; McCarthy, 1997; Morreau and Kraus, 1998; Perlis, 1985; 1988; des Rivières and Levesque, 1988; Turner, 1990]. A couple of the most successful recent frameworks using the predicate approach in formalising the knowledge and belief of agents can be found in Morreau and Kraus [1998], Grant *et al.* [2000] and Fasli [2003]. Since the predicate approach is the one we are going to pursue in this thesis, we will briefly review some of the most common arguments given in favour of it. This is done in the following subsections.

2.4.1. Logical Omniscience. In the classical operator approach, one has a language of propositional modal logic with K and B being modal operators. This language is given a possible-world semantics. The possible-world semantics seems very suitable and intuitively appealing as a semantics for languages involving knowledge and belief. The idea is that for $K\varphi$ to be true, the proposition expressed by φ must be true in every world that the agent in question considers possible. For the agent to know something is thus for this something to hold in all possible states of affairs that are indistinguishable to the agent from the actual state. Let us give an example. Imagine that the dotted block in Figure 2.1 is blocking the view of the agent such that it cannot see the white block. Then the agent would both consider the world in which there *is* a white block on the floor behind the dotted block and the world in which there is *not* as possible (since it is aware that there might be something on the floor behind the dotted block that it cannot see). Thus in this case the sentence $K\,On(\mathit{white}, \mathit{floor})$ would *not* be true in the possible-world semantics. But we would still have that the sentence $K\,On(\mathit{dotted}, \mathit{black})$ is true, since the agent does not consider any worlds possible in which $On(dotted, black)$ is not the case.

Although the possible-world semantics seems intuitively appealing and has several advantages, it also possesses some major disadvantages. One of these is the *logical omniscience problem*. On the possible-worlds account, $K\varphi$ is true if φ is true in all worlds considered possible by the agent. Suppose φ and ψ are logically equivalent formulas. Then they must receive the same truth-value in all worlds. In particular, they must receive the same truth-value in all worlds considered possible by the agent. Suppose furthermore that $K\varphi$ holds. Then φ is true in all worlds considered possible by the agent, and by the logical equivalence of φ and ψ, the formula ψ must be true in all of these worlds as well. This implies that $K\psi$ holds. Thus, if the agent knows φ and if furthermore φ and ψ are logically equivalent, then the agent necessarily knows ψ as well. This property is called *logical omniscience*. It is the property that knowledge is closed under logical equivalence.

Logical omniscience seems as an unreasonable property of artificial intelligence agents (and humans, for that matter) to have. If I know φ, and if φ and ψ are logically equivalent, I do *not* necessarily know ψ, because I might be unaware that φ and ψ are indeed equivalent. For this reason, many have considered the combination of the operator approach and the possible-world semantics as an improper framework for formalising knowledge and belief. It should be noted, however, that if we think of $K\varphi$ as expressing that φ is *implicitly known*, then logical omniscience might be an acceptable property after all.

The predicate approach does not suffer from the logical omniscience problem. In the standard predicate approach, one has a language of first-order predicate logic with K and B being one-place predicate symbols. This language is given a standard Tarskian (truth-functional) semantics, which means that we assign an *extension* to every predicate symbol in the language. The idea is here that for $K\varphi$ to be true, φ must be in the extension of the predicate symbol K (or rather, the *name* of φ must be in this extension). In this semantics, there is nothing forcing $K\varphi$ to be semantically related to $K\psi$ even if φ and ψ are logically equivalent. Whether $K\varphi$ and $K\psi$ are true or false only depends on whether φ and ψ are included in the extension of K or not, and this extension can be chosen independently of the meaning we assign to φ and ψ. We can always choose to include φ in the extension but exclude ψ. Thus, we are not forced to have logical omniscience.

One of the reasons put forward to prefer the predicate approach over the operator approach is that logical omniscience can be avoided [Asher and Kamp, 1986; Davies, 1990; Fasli, 2003; Turner, 1990; Whitsey, 2003]. More generally, we can see from the presentation of the two approaches given above that the predicate approach offers a much more *fine-grained* notion of proposition than the operator approach. Note, however, that there has also been considerable work on how to avoid the logical omniscience problem while staying within the operator approach. One of the possibilities is to introduce *impossible worlds*. This and other approaches to avoid the problem are reviewed in Fagin *et al.* [1995].

2.4.2. Introducing New Modalities. McCarthy [1997] gives several arguments for the inadequacy of modal logic as the basis for constructing artificial intelligence agents. One of his arguments is that modal logic is unsuitable to take care of the situation in which an agent adds a new *modality* to its language. By *modality* in this context is meant *a manner in which an agent (or group of agents) relates to a proposition or a sentence*. Such modalities are also known as *propositional attitudes*. As classical examples of such modalities we have *knowing* and *believing*. *Knowing* is a modality, since it is used to express the certain way in which an agent relates to a proposition (sentence) when it is said to know this proposition (sentence). As further examples of modalities (propositional attitudes) we have: intending, fearing, hoping, desiring, obligating. Concerning such modalities, McCarthy [1997] writes: "Human practice sometimes introduce new modalities on an *ad hoc* basis. [...] Introducing new modalities should involve no more fuss than introducing a new predicate. In particular, human-level AI requires that programs be able to introduce modalities when this is appropriate."

The idea is that an agent might come to need a new modality in the course of its reasoning, and the formalism with which the agent is built should somehow support this. For instance, an agent might come to need a modality for expressing *common knowledge* or *distributed knowledge* among a group of agents to carry out the required reasoning (an example of this is the reasoning required in the *muddy children puzzle* [Fagin *et al.*, 1995]). The agent should somehow have access to such modalities, even though it has not explicitly been equipped with these from the beginning. On the operator approach in general—and in modal logic in particular—only modalities that are included in the language from the beginning will be accessible to the agent. If we want agents to be able to reason about common and distributed knowledge on the operator approach, we must have operators for these modalities in our language as well as a number of axioms in the underlying theory expressing the basic properties of these modalities. The occasional need for new modalities might thus lead to a proliferation of theories on the operator approach, since we need to extend the language and add corresponding axioms to the underlying theory whenever we wish to be able to handle a new modality.

As we will show in Section 3.5.2 and Section 6.2.2, the predicate approach fares much better with respect to this problem. In this approach, modalities such as common and distributed knowledge can be *reduced* to the basic modality of knowing. This means that the language only needs to be equipped with a knowledge modality (a K predicate), and that other modalities such as common and distributed knowledge can be expressed entirely in terms of this basic modality. Other modalities one might come to need and which on the predicate approach can be reduced to the basic modality of knowing are: *knowing more*, *knowing about*, *only knowing*, *knowing at least*, *knowing at most* (the last three of these modalities are considered in Levesque and Lakemeyer [2000]). The reason that these modalities can be reduced to knowledge in the predicate approach but not the operator approach is that the predicate approach is much more *expressive* than the operator approach. Knowledge can be referred to in much more complex ways in the predicate approach than the operator approach.

The conclusion—which we share with McCarthy [1997]—is that since we would like to allow our artificial intelligence agents to add new modalities to their languages whenever needed, we should choose the predicate approach rather than the operator approach in the formalisation of these agents' knowledge.

2.4.3. Expressive Power. Probably the strongest argument in favour of the predicate approach is that it gives a much more *expressive* formalism than can be obtained with the operator approach (at least as long as we do not consider higher-order formalisms). This argument in favour of the predicate approach has been put forward by Asher and Kamp [1986], Carlucci Aiello *et al.* [1995], Davies [1990], Fasli [2003], Moreno [1998], Morreau and Kraus [1998], des Rivières and Levesque [1988], Turner [1990] and Whitsey [2003]. It constitutes the main reason for choosing the predicate approach in this thesis. There are simply propositions that cannot be expressed on the operator approach that are essential to have in a formalism that claims to support strong introspection. We gave a number of examples of such propositions in Section 1.3.2. We will now give some more, and explain why these propositions are important to be able to express.

One of the simplest examples of a sentence expressible in the predicate approach but not the operator approach is

$$\text{“Bill knows everything Sue knows”} \tag{2.2}$$

(this example appears in Asher & Kamp [1986] and Attardi & Simi [1995]). The sentence can be formalised in the predicate approach as

$$\forall x \left(K_{sue}\, x \rightarrow K_{bill}\, x \right).$$

The reason that the sentence can not be formalised in the operator approach is that it involves quantifying over knowledge. In the predicate approach such quantifications can be expressed by applying K to a variable x and then quantify over this variable. This is not possible if K is an operator, however, since then K only applies to well-formed formulas and not to terms such as the variable x. The problem could possibly be solved in the operator approach by moving into a *higher-order modal logic*, but this possibility has not been investigated by the author. Higher-order modal logics are not very well-known although several such logics have been developed [Fitting, 2002b; Gallin, 1975].

Why do we even want to be able to express a sentence such as (2.2)? Unless we can imagine agents that would gain from being able to express and reason about such a sentence, the fact that the predicate approach has higher expressive power will not be a strong argument for choosing that approach. But it is quite easy to imagine situations involving several agents (*multi-agent systems*) in which a sentence such as (2.2) is required. We give an example of this in the following.

Example 2.1 (Knowing more)**.** Imagine a world in which there are three agents: Bill, Sue and Ole (the third one is a Danish agent). We will consider these agents to be software agents "living" on the Internet—however, the example could just as well have been made with the agents being for instance mobile robots. Bill and Sue are agents assisting Ole in carrying out various information gathering tasks on the behalf of a human user. Suppose Ole asks the two agents Bill and Sue to find information on the Internet concerning *cool jazz*. They will do this by traveling around on the Internet and possibly communicating with other agents as they move along. When the two agents "return", Ole asks them where they have been. From their reports, Ole is able to infer that everywhere Sue has been, Bill has also been. From this fact Ole should be able to infer that Bill knows at least as much as Sue knows (about cool jazz). Thus we expect the following sentence to enter the knowledge base of Ole:

$$K_{ole} \forall x \, (K_{sue} \, x \rightarrow K_{bill} \, x)$$

or maybe rather:

$$K_{ole} \forall x \, (About\,(x, cool\ jazz) \wedge K_{sue} \, x \rightarrow K_{bill} \, x)\,.$$

This can only happen on the predicate approach, since only on the predicate approach can we express these two sentences. Coming to know that Bill knows everything Sue knows allows Ole to deduce that he only needs to ask Bill for information on the subject of cool jazz. He can thus immediately send Sue out to solve a new task while “questioning” only Bill about what he has learned about the subject.

An advocate for the operator approach might object to the necessity of the predicate approach in this example by noting that we could just introduce a 'knows more' modal operator into the operator language and then express the propositions above through this new operator. This would of course be possible, but it would leave us in the situation we argued against in Section 2.4.2: it should not be necessary to extend the entire underlying logical framework just to be able to express a piece of reasoning involving a new modality. In particular not when this new modality in a simple way can be derived from the basic modality of knowing.

The example above does not involve introspection, but the following simple variant of it does.

Example 2.2 (Knowing more)**.** We consider again the agent scenario presented above. Suppose that this time Bill and Sue are independently send out by two different users to gather information on cool jazz. They coincidentally meet on the Internet, and Bill asks Sue whether she knows anything about the subject. Note at this point that even to express that Sue knows *something* about cool jazz, we need the predicate approach:

$$\exists x \left(\mathit{About}(x, \mathit{cool\ jazz}) \wedge K_{\mathit{sue}}\, x \right).$$

When Sue answers yes to Bill's question, he asks her at which sites she has been obtaining her knowledge on the subject, and he realises that she has not been anywhere that he himself has not. From this he comes to know that he knows more about the subject than Sue:

$$K_{\mathit{bill}} \forall x \left(\mathit{About}(x, \mathit{cool\ jazz}) \wedge K_{\mathit{sue}}\, x \rightarrow K_{\mathit{bill}}\, x \right).$$

This is a piece of introspective knowledge, since it is knowledge held by Bill concerning Bill's own knowledge. Whether it is a piece of self-referential knowledge as well depends—as we will see in Example 6.10—on how we formalise the *About* predicate.

A couple of other examples of sentences only expressible in the predicate treatment are the following:

- “John knows that Bill knows something that John himself doesn't”:

  $$K_{\mathit{john}} \exists x \left(K_{\mathit{bill}}\, x \wedge \neg K_{\mathit{john}}\, x \right)$$

  (presented in [Priest, 1991; Attardi and Simi, 1995]).
- “John believes that he has a false belief”:

  $$B_{\mathit{john}} \exists x \left(B_{\mathit{john}}\, x \wedge \neg \mathit{True}(x) \right) \tag{2.3}$$

  (presented in [Perlis, 1985; Attardi and Simi, 1995]).

Furthermore, we need the predicate approach if we want to express sentences concerning knowledge shared by all agents in the world or all agents in a certain group.

Example 2.3 (Everybody knows)**.** To express that all agents know that the dotted block is on the black block, we can write

$$\forall x K_x \, On\,(dotted, black)\,. \tag{2.4}$$

This sentence is actually an abbreviation for

$$\forall x K\,(x, \ulcorner On\,(dotted, black)\urcorner)\,.$$

We will explain the details of these syntactic matters in Chapter 3. If we knew the names of all agents, we could replace the sentence (2.4) by a finite conjunction of sentences of the form

$$K_{N.N.} \, On\,(dotted, black)\,, \tag{2.5}$$

where N.N. is the name of a particular agent. Such a finite conjunction would be expressible in the operator approach, but it is not always likely that we know the names of all agents. In particular, what we might intend to express by (2.4) is that *any* agent at *any* time knows the dotted block to be on the black block. Since the number of agents might vary over time, there will be no finite conjunction of sentences on the form (2.5) that can replace the sentence (2.4). The sentence (2.4) is only expressible in the predicate approach. The reason is that in the operator approach one has an individual modal operator for each agent, so it is only possible to refer to the knowledge of a group of agents by referring directly to the names of the agents in this group. As before, the problem can be circumvented in the operator approach by introducing a new modality, but as we have argued in Section 2.4.2 this is an inadequate solution.

We have now given a number of examples of propositions expressible only on the predicate approach. It is furthermore easy to see that any proposition expressible on the operator approach is also expressible on the predicate approach: If φ is an operator sentence we simply have to replace all subformulas on the form $K\psi$ (or $B\psi$) by $K(\ulcorner\psi\urcorner)$ (or $B(\ulcorner\psi\urcorner)$) in order to get a corresponding predicate sentence (the details of this translation are carried out in des Rivières and Levesque [1988]). Thus we have succeeded in showing that the predicate approach is more expressive than the operator approach. This fact constitutes an important argument in favour of the predicate approach. Furthermore, from the examples we have given, we see that the predicate approach is needed in expressing many natural propositions involving knowledge—propositions we would like our agents to be able to express. In particular, the predicate approach allows us to quantify over knowledge, which is needed in strong introspection—for instance when expressing the introspective awareness that one has a false belief (sentence (2.3) above). This is our main reason for choosing the predicate approach in this thesis.

2.4.4. Concluding Remarks. Above we have given a number of arguments in favour of the predicate approach, but none in favour of the operator approach. If the predicate approach is so much superior to the operator approach, why is the operator path still the one most often taken? A number of both technical and philosophical reasons have been given in the literature. Among the technical reasons is that propositional modal logic is decidable whereas first-order predicate logic is only semi-decidable. Semi-decidability is, however, probably the price we have to pay to use any kind of logic as strongly expressive as predicate logic (and in this thesis we *need* such an expressive logic to be able to express strongly introspective beliefs, which a propositional modal logic cannot). Among the philosophical reasons is that it has been argued that the proper object of a belief is a *proposition* rather than a *sentence*, and only the operator approach with a possible-world semantics is faithful to this view. But the most important argument against the predicate approach is that it is prone to *inconsistency* [Kaplan and Montague, 1960; Montague, 1963; Thomason, 1980]. We already saw this in Chapter 1 with the *knower paradox* (Sections 1.4.1 and 1.4.2). If our ambition is to build artificial intelligence agents with strong introspective abilities we need the strongly expressive framework, and thus we are required to find a way to overcome the inconsistency problems. This is, as mentioned several times, what this thesis is all about.

Concluding this section of comparing the operator approach to the predicate approach, we should note that some authors refer to the operator approach as the *semantic approach* (or the *semantic treatment*) and the predicate approach as the *syntactic approach* (or the *syntactic treatment*). We will avoid using the latter terms in this thesis, since the semantic/syntactic distinction most often carries a somewhat different meaning: The semantic approach is used to describe situations in which the objects of knowledge and belief are taken to be *propositions*, and the syntactic approach is used to describe situations in which these objects are *sentences*. In most cases the operator/predicate distinction and the semantic/syntactic distinction coincide, but not always.

2.5. Aspects not Covered in the Thesis

There are several important aspects involved in the construction of artificial intelligence agents that we do not cover in this thesis. For instance we do not discuss *feasibility* and *tractability*, that is, whether we can implement inference engines for our logics that will be able to deduce useful conclusions from the facts in the knowledge base in reasonable time (useful conclusions *will* be deducible because of the semi-decidability of the logical theory constituting the knowledge base, but in practice we might face problems of not obtaining the desired conclusions as fast as we would like). We also do not discuss details of *implementation*, that is, whether our inference engine should be build on a first-order theorem prover or a suitable inference engine for extended logic programs.

In addition, we do not cover issues such as how to combine knowledge bases of agents with *actions* like moving blocks or asking questions. Introducing actions into the framework can be done for instance by using *knowledge based programs* [Fagin *et al.*, 1995]. These are programs that can be built on top of any formalism for representing and reasoning about knowledge. The formalism for reasoning about knowledge does therefore not itself need to have an ability to deal with actions. We are thus not required to include mechanisms for dealing with actions in our basic epistemic framework, but can think of these mechanisms as being something added on top of the completed epistemic framework.

Our logical theories are conceptually reasonably clean and simple by only being directly concerned with the two basic modalities *knowledge* and *belief*, and not for instance with *intentions*, *desires* and *obligations*. However, the consistency results we will be proving could also be applied to show the consistency of the predicate approach in formalising such modalities, at least to the extend that these modalities can be described by axiomatic principles similar to the principles used for knowledge and belief (see Section 3.7). Our logical theories are not directly equipped with capabilities allowing an agent to retract beliefs from its knowledge base if it discovers that some of these beliefs are mistaken or express propositions that no longer hold (the introduction of such capabilities are studied within the areas of *belief revision* and *belief update*, respectively). Finally, our formal framework supports neither *probabilistic* nor *temporal* representation and reasoning. Probabilistic representation and reasoning might be needed to properly formalise agents that make inaccurate observations and can be uncertain about their own observations and reasoning. Temporal representation and reasoning is needed if we want to add a time parameter to the knowledge an agent ascribes to itself and the other agents in its world, that is, if pieces of knowledge are formalised as being pieces of knowledge held at particular points in time.

All these simplifications of the subject matter—of which we have mentioned far from all—are deliberate. The problem with which this thesis is concerned—the problem of making a proper formalisation of strong introspection of agents—is theoretically very challenging. We therefore choose to study the problem in the simplest context possible: a context involving no *actions*, *communication*, *intentions*, *belief revision*, etc. By studying the problem of introspection somewhat in isolation from these other issues, we are given the best and cleanest point of departure for attacking the problem, and it makes the entire treatment considerably more clear. We have no interest in adding complicating factors to our framework, such as actions or further basic modalities, before the fundamental problem of introspection has been solved. We are aware that taking more factors into account in some cases actually *can* make the problems involved in introspection easier to circumvent: adding an ability to revise one's knowledge base might for instance be a way to recover the knowledge base from inconsistency after having engaged in paradoxical introspective reasoning such as involved in the knower paradox (Section 1.4.2). However, even if one prefers to seek to avoid the problems of self-reference and introspection by *extending* the underlying logical framework, the problem in its more pure form is still of interest. There is no doubt that recovering from an inconsistency should be treated differently depending on whether the inconsistency is a result of a mistake made by the agent or a result of paradoxical reasoning. We would even like the agent itself to be able to figure out which one is the case in a given situation. Thus we still need to investigate the problem of introspection in its more pure form to be able to characterise the types of classical logical reasoning involving introspection that can lead to paradoxes.

Much of the recent research in logic-based artificial intelligence and knowledge-based systems goes in the direction of adding new dimensions to the representing and reasoning frameworks: adding *time*, *contexts*, *desires*, *intentions*, *speech acts*, etc. As mentioned, this thesis tries to solve a problem that appears already in formalisms not incorporating any of these additional dimensions, and for that reason we choose to stay within the more 'pure' formalisms.

CHAPTER 3

First-Order Predicate Logic for Knowledge and Belief

In this chapter we will give a detailed definition of the logical theories to be used in the thesis, and discuss their fundamental mathematical properties. This involves introducing most of the basic technical machinery on which the entire thesis relies. A considerable effort has been put into choosing the exact right definitions of our basic concepts. This has been done primarily with the aim to make the proofs of our results concerning these concepts as simple as possible. A consequence of this is that some of the definitions are slightly more involved than would have otherwise been the case. This implies that the reader may have to invest somewhat more in reading this chapter introducing the basic concepts, but the extra effort greatly pays off in the later and more advanced chapters, since our choices of definitions result in considerable simplifications.

To formalise the knowledge and belief of agents we will be using languages and theories of first-order predicate logic. More precisely, the knowledge base of an agent will always be a theory S in a language L of first-order predicate logic. The objects L and S should satisfy certain requirements to make sense as being, respectively, the language and knowledge base of an agent. In this chapter we will define these requirements, and discuss the basic properties of these languages and theories. After that, we will define a number of axiom schemes called *reflection principles*. These are general logical principles by which an agent can infer consequences of its own knowledge. Such reflection principles play a central role in the construction of logic-based agents.

3.1. First-order Predicate Logic

We begin by introducing a number of concepts and standard notions related to first-order predicate logic. The reader is assumed to be familiar with this subject, so the following is just meant as a brief summary as well as to make precise the exact version of the logic we are using here.

We take the *connectives* of first-order predicate logic to be $\neg$, $\wedge$ and $\forall$. When using $\vee$, $\rightarrow$, $\leftrightarrow$ and $\exists$ in formulas, these formulas are simply abbreviations of formulas containing only $\neg$, $\wedge$ and $\forall$. The *variables* of first-order predicate logic are

$$x_0, x_1, x_2, x_3, \ldots$$

We sometimes use x, y and z to denote variables among $x_0, x_1, \ldots$ The connectives and variables together with the following three symbols

$$, \quad) \quad ($$

are called the **logical symbols** of first-order predicate logic. In addition to the logical symbols, a language of first-order predicate logic contains a number of *non-logical symbols*. These are *constant symbols*, *function symbols* and *predicate symbols*. We assume all non-logical symbols to be taken from the following fixed vocabulary, where n ranges over the positive integers

- Constant symbols: $a_1, a_2, a_3, \ldots$
- n-place predicate symbols: $A_1^n, A_2^n, A_3^n, \ldots$
- n-place function symbols: $f_1^n, f_2^n, f_3^n, \ldots$

These symbols together form the **non-logical symbols** of first-order predicate logic. In practice, however, we will allow any string of lower-case Latin letters—such as *dotted* or *neg*—to be the name of a constant or function symbol. These strings should then be considered as simply being notational variants of symbols belonging to the fixed vocabulary above. Similarly, we will use capitalised strings of Latin letters—such as *On* and *About*—as predicate symbols. Predicate symbols are capitalised to distinguish them from function symbols. By a **formula of first-order predicate logic** we understand any well-formed formula constructed from the logical and non-logical symbols above. We will make the assumption that any variable x in a formula φ is quantified at most once, that is, for any variable x there is at most one occurrence of $\forall x$ (or $\exists x$) in φ.

The reason we choose a fixed vocabulary of constant, function and predicate symbols is primarily that it allows us to construct a *Gödel numbering* in which a unique natural number is associated with any expression of *any* first-order language. This has the following advantage. Assume we work with some language L of first-order predicate logic and a Gödel numbering g. The mapping g is assigning unique numbers to all expressions in L. At some point we might need to extend the language L with new constant, function or predicate symbols. With a fixed vocabulary to which all such new symbols belong, we can from the beginning choose g such that it is also giving Gödel numbers to the expressions of the extended language. We do therefore not need to extend the Gödel numbering every time we choose to extend the first-order language we are considering. Gödel numberings are treated in detail in Section 3.3.

Languages of first-order predicate logic will most often simply be called **first-order languages**. First-order languages are identified with their sets of *sentences*. Thus if L denotes a first-order language and we want to express that φ is a sentence in L, we can simply write $\varphi \in L$. A **sentence** in a first-order language is a *closed formula*, that is, a formula with no free variables. Formulas with at least one free variable are called *open formulas*. Terms without variables are called **closed terms**. In logic programming, closed terms have been rebaptised as **ground terms**. We will use both names in this thesis. The set of closed terms of a first-order language L is denoted $\mathrm{Terms}(L)$. Formulas without connectives are called **atomic formulas**. We will generally use lower-case Greek letters to denote formulas and sentences of first-order languages (most often φ,ψ,α,β and γ).

We will allow some abbreviation of formulas in first-order languages. As usual, we will omit parentheses whenever this is not likely to cause confusion. We will even, when possible, allow the omission of parentheses around the arguments of one-place function and predicate symbols. Thus if K is a one-place predicate symbol and τ is a term, we will allow the formula $K(\tau)$ to be abbreviated as $K\tau$. If s is a one-place function symbol, we will allow the term $s(s(x))$ to be abbreviated ssx.

A first-order language is said to have a **finite vocabulary** if it only contains finitely many constant symbols, function symbols and predicate symbols. An example of a language with a finite vocabulary is the **language of arithmetic**, which contains the following set of non-logical symbols

- A single two-place predicate symbol '$=$'.
- A single constant symbol '0'.
- A one-place function symbol 's'.
- Two three-place predicate symbols '$+$' and '$\cdot$'.

Our definition of the language of arithmetic is slightly non-standard. Usually $+$ and $\cdot$ are taken to be two-place function symbols rather than three-place predicate symbols. We choose the non-standard definition, since it allows us to simplify a number of the definitions and proofs given in the thesis. We will also use the symbols $>$, $<$, $\geq$ and $\leq$ in expressing formulas within the language of arithmetic, but as usual we consider formulas involving these symbols as abbreviations of formulas using only the symbols mentioned above. When we say that a language L *contains* the language of arithmetic, we mean that among its non-logical symbols it has the ones mentioned above.

We take **first-order predicate logic** to consist of the following logical axioms [Mendelson, 1997]:

P1. $\alpha \rightarrow (\beta \rightarrow \alpha)$
P2. $\alpha \rightarrow (\beta \rightarrow \gamma) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$
P3. $(\neg\alpha \rightarrow \neg\beta) \rightarrow ((\neg\alpha \rightarrow \beta) \rightarrow \alpha)$
P4. $\forall x_i \alpha(x_i) \rightarrow \alpha(\tau)$, if τ is free for x_i in $\alpha(x_i)$.
P5. $\forall x_i (\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \forall x_i \beta)$, if α contains no occurences of x_i.

These are *axiom schemes* where α and β are ranging over the formulas of first-order predicate logic. **First-order predicate logic with equality** contains in addition the following logical axioms called the *axioms of equality*

P6. $x_1 = x_1$
P7. $x = y \rightarrow (\alpha(x, x) \rightarrow \alpha(x, y))$, where x and y are any variables such that y is free for x in $\alpha(x, x)$, and $\alpha(x, y)$ arises from $\alpha(x, x)$ by replacing some, but not necessarily all, free occurences of x by y.

REMARK. *Throughout this thesis, by first-order predicate logic we mean first-order predicate logic with equality. We will often simply refer to it as predicate logic.*

Let L be a first-order language. A **theory** in L is simply a set of sentences in L. These sentences are called the **(non-logical) axioms** of the theory. Suppose S is a theory in L and φ is a formula in L. We say that φ is a **theorem** in S or that φ is **provable** in S if there is a formal proof of φ employing only

- The logical axioms $P1$–$P7$ instantiated with formulas of L.
- The non-logical axioms in S.
- The inference rules MP (modus ponens) and Gen (generalisation).

We write $S \vdash \varphi$ to express that φ is a theorem in S. Formal proofs will be written in the style of Mendelson [1997]. By a **first-order theory** we understand a theory in any first-order language.

An example of a theory in the language of arithmetic is **Robinson's Q**, which contains the following non-logical axioms

Q1. $\exists! x_3 (+(x_1, x_2, x_3))$ [1]
Q2. $\exists! x_3 (\cdot(x_1, x_2, x_3))$
Q3. $x_1 = x_2 \rightarrow sx_1 = sx_2$
Q4. $x_1 = x_2 \wedge +(x_1, x_3, x_4) \wedge +(x_2, x_3, x_5) \rightarrow x_4 = x_5$
Q5. $x_1 = x_2 \wedge +(x_3, x_1, x_4) \wedge +(x_3, x_2, x_5) \rightarrow x_4 = x_5$
Q6. $x_1 = x_2 \wedge \cdot(x_1, x_3, x_4) \wedge \cdot(x_2, x_3, x_5) \rightarrow x_4 = x_5$
Q7. $x_1 = x_2 \wedge \cdot(x_3, x_1, x_4) \wedge \cdot(x_3, x_2, x_5) \rightarrow x_4 = x_5$
Q8. $sx_1 = sx_2 \rightarrow x_1 = x_2$
Q9. $\neg 0 = sx_1$

[1] As usual, for any formula $A(x)$ we write $\exists! x A(x)$ as an abbreviation for the formula $\exists x A(x) \wedge \forall y \forall z (A(y) \wedge A(z) \rightarrow y = z)$.

Q10. $\neg x_1 = 0 \rightarrow \exists x_2\,(x_1 = sx_2)$
Q11. $+(x_1, 0, x_1)$
Q12. $+(x_1, x_2, x_3) \rightarrow +(x_1, sx_2, sx_3)$
Q13. $\cdot(x_1, 0, 0)$
Q14. $\cdot(x_1, sx_2, x_3) \wedge \cdot(x_1, x_2, x_4) \wedge +(x_4, x_1, x_5) \rightarrow x_3 = x_5$

Since our language of arithmetic is non-standard with $+$ and $\cdot$ being predicate symbols rather than function symbols, these axioms of Robinson's system Q are also non-standard. They are obtained from the standard axioms by applying the transformation introduced in Section 2.9 of Mendelson [1997]. The theory Q will play an important role in the thesis. The central property of Q is that all recursive functions and relations are representable in it (see Section 3.5).

A theory S in a first-order language L is called **finitely axiomatisable** if there is a theory V containing only finitely many non-logical axioms and having the same theorems as S. The theory S is called **recursively axiomatisable** if its set of non-logical axioms is recursive.

3.2. First-order Agent Languages and Theories

We are now in a position to define the requirements we will put on our languages and theories to be used in formalising the knowledge and belief of agents. Languages and theories satisfying the requirements will be called *first-order agent languages* and *first-order agent theories*, respectively.

Definition 3.1. *Let L be a language of first-order predicate logic. L is called a* ***(first-order) agent language*** *if it satisfies the following requirements*

(i) *L contains the language of arithmetic.*
(ii) *L contains a two-place predicate symbol denoted K.*

Definition 3.2. *Let L be an agent language and let S be a theory in L. S is called a* ***(first-order) agent theory*** *in L if it satisfies the following requirements*

(i) *S extends the theory Q.*
(ii) *S is recursively axiomatisable.*

In the following sections we will explain the requirements we have put on agent languages and theories. Throughout we assume L to be a first-order agent language and S to be a first-order agent theory in L. The language L can be thought of as the language with which an agent is equipped to express its knowledge and belief and S can be thought of as this agent's knowledge base.

Our first requirement—item (i) in Definition 3.1—has to do with the ability of the agent to code sentences of its language as terms in this language. We explain this in detail in the following.

3.3. Coding of Sentences

To allow introspection, the agent has to be able to refer to the objects in its own knowledge base. These objects are sentences in the language L. The objects with which the agent refers are also sentences in L, so we need a way to allow sentences in L to refer to other sentences in that same language. This is done using a *Gödel coding* (*Gödel numbering*). Through a Gödel coding we can associate to each formula φ in L a unique term $\ulcorner\varphi\urcorner$ in L. The term $\ulcorner\varphi\urcorner$ is called the *Gödel code* of φ. When a formula ψ in L contains the Gödel code of another formula φ in L, we will think of ψ as *referring* to φ through this Gödel code.

Throughout this thesis we will be using a fixed Gödel coding, which we are now about to present. We present the coding in detail even though the reader is assumed already to be familiar with such codings. The primary reason for presenting the coding in detail is that there are certain properties that not all Gödel codings share, but that we need our coding to possess. We therefore need to be precise about our choice of coding.

Definition 3.3 (Mendelson [1997]). *Let M denote the union of the set of logical and non-logical symbols of first-order predicate logic. We define an injective map $g : M \to \mathbb{N}$ by, for all $k, n > 0$*

$$\begin{array}{ll} g(()=3 & g(x_k)=13+8k \\ g())=5 & g(a_k)=7+8k \\ g(,)=7 & g(f_k^n)=1+8(2^n3^k) \\ g(\neg)=9 & G(A_k^n)=3+8(2^n3^k) \\ g(\wedge)=11 & \\ g(\forall)=13. & \end{array}$$

Definition 3.4 (Mendelson [1997]). *Let M be as above and let $u_1 \cdots u_r$ be a string of symbols in M. The* ***Gödel number*** *of the expression $u_1 \cdots u_r$ is the natural number*

$$2^{g(u_1)}3^{g(u_2)}5^{g(u_3)} \cdots p_{r-1}^{g(u_{r-1})}p_r^{g(u_r)},$$

where p_j for all $j > 0$ denotes the j'th prime number.

Through this definition we get a unique Gödel number associated with each formula in each first-order language. But our goal was to associate terms and not numbers with formulas. The following definition shows how this can be done through the given Gödel numbering.

Definition 3.5. *Let M be as above. Let L be a first-order language containing the language of arithmetic and let ε denote a string of symbols from M. The* ***Gödel code*** *of the expression ε is the following closed term in L*

$$\underbrace{ss\cdots s}_{n}0$$

with n occurrences of s, where n is the Gödel number of ε. We denote this term by $\ulcorner\varepsilon\urcorner$. The string ε is said to be the expression ***denoted*** *by the term $\ulcorner\varepsilon\urcorner$, and $\ulcorner\varepsilon\urcorner$ is said to be the* ***name*** *of ε.*

Terms of the form $ss\cdots s0$ are usually called **numerals**. For every $n \in \mathbb{N}$, we use $\bar{n}$ as an abbreviation for the term $ss\cdots s0$ with n occurrences of s. We say that $\bar{n}$ is the numeral **denoting** the natural number n. When no confusion can occur, we identify numerals with the natural numbers they denote. That is, we identify $\bar{n}$ with n. This means that the *Gödel numbers* and the *Gödel codes* of expressions will be identified. For instance, suppose φ is a formula of predicate logic. Then we will use the notation $\ulcorner\varphi\urcorner$ both to denote the Gödel *number* of φ, which is a natural number, and to denote the Gödel *code* of φ, which is the corresponding numeral.

The above definition successfully associates a unique term in the language of arithmetic with every formula of predicate logic. Let L be a first-order language containing the language of arithmetic. Suppose L contains a one-place predicate symbol K. Each sentence φ in L has a Gödel code $\ulcorner\varphi\urcorner$, which is a term in L (a numeral). For any such sentence we will think of the sentence $K(\ulcorner\varphi\urcorner)$ as expressing that φ has the property expressed by the predicate K. If K expresses the property of being known by an agent, then $K(\ulcorner\varphi\urcorner)$ can be read as saying "φ is known by the agent". Similarly, $K\left(\ulcorner K\left(\ulcorner\varphi\urcorner\right)\urcorner\right)$ can be read as saying

"$K(\ulcorner\varphi\urcorner)$ is known by the agent"

or

"The sentence "φ is known by the agent" is known by the agent".

When no ambiguity can occur, we will in formulas allow the term $\ulcorner\varphi\urcorner$ to be abbreviated by φ. Thus $K(\ulcorner\varphi\urcorner)$ will be abbreviated $K(\varphi)$ or simply $K\varphi$, and $K\left(\ulcorner K\left(\ulcorner\varphi\urcorner\right)\urcorner\right)$ will be abbreviated $K\left(K(\varphi)\right)$ or simply $KK\varphi$. Using such abbreviations result in much simpler notation.

Requirement (i) in the definition of a first-order agent language (Definition 3.1) states that the language contains the language of arithmetic. This requirement is included to make sure that the agent has *names* (Gödel codes) for all expressions in its language. This allows it to refer freely to the objects in its knowledge base as needed to do introspection.

3.4. Predicates for Knowledge and Belief

Requirement (ii) in Definition 3.1 states that any first-order agent language contains a two-place predicate symbol K. As before, this predicate will be used to express *knowledge*. The first argument position is intended to contain the name of an agent. The intended interpretation of

$$K(john, \varphi) \tag{3.1}$$

is therefore that φ is known by the agent John. To simplify notation, we will most often write the first argument to K as a subscript of K, so that (3.1) becomes

$$K_{john}(\varphi)$$

or simply

$$K_{john}\,\varphi.$$

The fact that K is a *two*-place predicate symbol is required in connection with *multi-agent systems* where we want our agents to be able to express and reason about the knowledge of other agents in the system. To express that John knows that Sue knows that the dotted block is on the black block, we would write

$$K_{john} K_{sue}\, On\,(dotted, black)$$

which is an abbreviation for

$$K(john, \ulcorner K(sue, \ulcorner On\,(dotted, black)\urcorner)\urcorner).$$

When an agent wishes to express its *own* knowledge, we assume that the constant symbol 0 is put in the first argument position to K. That is, we think of 0 as being a default *name* for the agent itself. Thus if

$$K_0\, On\,(dotted, black)$$

is a sentence in the knowledge base of an agent, the intended interpretation is that this agent itself knows the dotted block to be on the black block. In this case we will often omit the subscript and simply write

$$K\, On\,(dotted, black)\,,$$

as we have been doing so far. To simplify notation, we will even allow to omit the subscript of K (the first argument to K) in all cases where the value of this subscript is not of central importance to the situation at hand.

We have not required agent languages to contain a predicate symbol B to express *belief*. The symbol B is left out to simplify notation. Our results will, however, be independent of whether K is intended to denote *knowledge* or *belief* (or any other modality for that matter). Thus we will sometimes use the predicate symbol K to denote *knowledge* and sometimes to denote *belief*. Treating only one modality at a time—which we always denote K—simplifies the exposition in a number of places. The simplification does not result in a loss of generality, since we can always construct agent languages and theories with more than one modality by adding one at a time. In examples, however, we will most often keep the notation we have been using so far: using K to denote knowledge and B to denote belief.

For any first-order agent language L, we will use $L - \{K\}$ to denote the language L with the predicate symbol K removed. Thus a theory in $L - \{K\}$ will be a theory in which there is no mention of knowledge (belief).

3.5. Representability

3.5.1. Definitions and Theorems. We have required that any first-order agent theory S extends the theory Q—this is requirement (i) of Definition 3.2. The primary reasons for this are that it ensures that the Gödel coding becomes well-behaved and that all recursive functions and relations become representable in S. Representability is defined in the following way.

DEFINITION 3.6 (After Mendelson [1997]). *Let S be a theory in a first-order language extending the language of arithmetic. Let R be an n-place relation over the natural numbers ($n > 0$). We say that R is **representable** in S if there is a formula $R(x_1, \ldots, x_n)$ in S with free variables $x_1, \ldots, x_n$ such that the following holds*

- *For all $k_1, \ldots, k_n \in \mathbb{N}$, if $\mathsf{R}(k_1, \ldots, k_n)$ is true then $S \vdash R(k_1, \ldots, k_n)$.*
- *For all $k_1, \ldots, k_n \in \mathbb{N}$, if $\mathsf{R}(k_1, \ldots, k_n)$ is false then $S \vdash \neg R(k_1, \ldots, k_n)$.*

*We say that the formula $R(x_1, \ldots, x_n)$ **represents** R in S.*

DEFINITION 3.7 (After Mendelson [1997]). *Let S be as above. Let a function $\mathsf{f} : \mathbb{N}^n \to \mathbb{N}$ be given ($n > 0$). We say that f is **representable** in S if there is a formula $F(x_1, \ldots, x_{n+1})$ in S with free variables $x_1, \ldots, x_{n+1}$ such that the following holds*

- *For all $k_1, \ldots, k_{n+1} \in \mathbb{N}$, if $\mathsf{f}(k_1, \ldots, k_n) = k_{n+1}$ then $S \vdash F(k_1, \ldots, k_{n+1})$.*
- *$S \vdash \exists! x_{n+1} F(x_1, \ldots, x_n, x_{n+1})$.*

*We say that the formula $F(x_1, \ldots, x_{n+1})$ **represents** f in S.*

We have the following well-known results regarding representability of recursive functions and relations.

THEOREM 3.8 (After Mendelson [1997]). *Let R be a recursive n-place relation ($n > 0$). There exists a formula $R(x_1, \ldots, x_n)$ in the language of arithmetic which represents R in any theory extending Q. In particular, the formula $R(x_1, \ldots, x_n)$ represents R in any first-order agent theory.*

PROOF. If R is recursive, we know that there is a formula $R(x_1, \ldots, x_n)$ representing R in the theory Q [Mendelson, 1997]. From this it immediately follows that $R(x_1, \ldots, x_n)$ must also represent R in any theory extending Q. Since by requirement (i) of Definition 3.2, any first-order agent theory extends Q, the formula $R(x_1, \ldots, x_n)$ must in particular represent R in any such theory. $\square$

Theorem 3.9 (After Mendelson [1997]). *Let $\mathsf{f} : \mathbb{N}^n \to \mathbb{N}$ be a recursive function ($n > 0$). There exists a formula $F(x_1, \ldots, x_{n+1})$ in the language of arithmetic which represents f in any theory extending Q. In particular, $F(x_1, \ldots, x_{n+1})$ represents f in any first-order agent theory.*

Let R be a recursive n-place relation over the natural numbers. By Theorem 3.8, there is a *single* formula $R(x_1, \ldots, x_n)$ representing R in *every* theory extending Q, and thus in particular in *every* first-order agent theory. We can therefore simply refer to this formula as *the formula representing* R, without referring to any particular theory. Given a relation R, we will therefore often simply say "let $R(x_1, \ldots, x_n)$ be a formula representing R" rather than "let $R(x_1, \ldots, x_n)$ be a formula representing R in the theory S". The same will be done with formulas representing recursive functions.

3.5.2. Examples. Theorem 3.8 demonstrates an important property of first-order agent theories, since it allows our agents to express statements concerning any recursive set of sentences. Suppose S is a first-order agent theory in a language L. If M is a recursive set of sentences in L, then by Theorem 3.8 there is a formula $M(x_1)$ representing the set of Gödel numbers of the elements in M. The following sentence

$$\forall x \, (Mx \to Kx)$$

will therefore be a sentence that the agent can use to express that all the sentences in M are known. In the following we will give some examples of how this can be used.

Example 3.10 (The *About* predicate). We have a couple of times been considering an *About* predicate (sections 1.3.2 and 2.4.3). The intended interpretation of

$$\mathit{About}\,(\tau_1, \tau_2)$$

is that τ_1 denotes a formula which expresses a proposition involving the object denoted by τ_2. For instance, we would say that the sentence "snow is white" is *about* snow, so we intend the sentence

$$\mathit{About}\,(\mathit{Is}\,(\mathit{snow}, \mathit{white})\,, \mathit{snow})$$

to hold. Conversely, we do not intend

$$\mathit{About}\,(\mathit{Is}\,(\mathit{snow}, \mathit{white})\,, \mathit{cool\ jazz})$$

to hold. Using the fact that all first-order agent theories can represent all recursive relations makes it easy to give a good formalisation of the *About* predicate in such theories. Let About be the following binary relation over the natural numbers

$$\mathsf{About} = \{(\ulcorner\varphi\urcorner, \ulcorner c\urcorner) \mid \varphi \text{ is a sentence and } c \text{ is a constant symbol occurring in } \varphi\}.$$

This is obviously a recursive relation.[2] There must therefore exist a formula $About(x_1, x_2)$ representing the relation About in any first-order agent theory S. Let S be any such theory and suppose it is the knowledge base of an agent. Using the definition of the relation About we get the following theorems in S

$$\begin{aligned}
&S \vdash About\,(On\,(striped, dotted)\,, striped)\\
&S \vdash About\,(On\,(dotted, striped)\,, striped)\\
&S \vdash About\,(\neg On\,(striped, dotted)\,, striped)\\
&S \vdash \neg About\,(On\,(black, dotted)\,, striped)\,.
\end{aligned} \tag{3.2}$$

We see that these theorems are consistent with the intended interpretation of the *About* predicate. Given this formalisation of the *About* predicate, the agent will be able to properly express that it has no knowledge about a striped block:

$$K\forall x\,(About\,(x, striped) \rightarrow \neg Kx)\,. \tag{3.3}$$

If the sentence (3.3) is a theorem in S and if we are given the appropriate axioms to allow the agent to infer consequences of its knowledge (see Section 3.7 below), then from (3.3) it will be able to infer for instance

$$K\,(\neg K\,On\,(striped, dotted) \wedge \neg K \neg On\,(striped, dotted))\,,$$

expressing that it does not know whether the striped block is on the dotted block or not.

Actually, the formalisation of *About* we have given turns out to present some problems related to introspection and self-reference. We will present these problems in Example 4.1 and a solution to them in Example 6.10. The purpose of the example given has simply been to illustrate that representability of recursive relations is quite "handy" for an agent to have when expressing knowledge or ignorance of a set of sentences.

[2]We will not be giving detailed proofs of the recursiveness of any functions or relations in this thesis. In the cases we consider, it will always be fairly obvious that the functions and relations in question are computable. To get from computability to recursiveness we rely on *Church's Thesis*.

Example 3.11 (Formalising common knowledge)**.** Representability of recursive relations in first-order agent theories also gives us the possibility of formalising *common knowledge* in these theories without the need to introduce extra non-logical symbols or axioms (cf. the discussion in Section 2.4.2). The concept of common knowledge can be described in the following way. Assume we are given a group of agents and a sentence φ. For the group to have *common knowledge* of φ means that: everyone in the group knows φ, everyone in the group knows that everyone knows φ, everyone in the group knows that everyone knows that everyone knows φ, and so on [Fagin *et al.*, 1995]. Common knowledge is a considerably stronger modality than "everyone knows". Consider the following example.

In the Hans Christian Andersen fairy-tale "The Emperor's New Clothes", the Emperor appears naked in front of a large group of people. They all notice that he is naked, so it would be reasonable to say that *everyone knows* the emperor is naked. However, the group of people does not have *common knowledge* of the fact that the Emperor is naked, since they are not sure about what the other people see. Therefore, if A and B are different persons, then A does *not* know that B knows that the Emperor is naked. That is, not until the little child says: "But the Emperor has nothing at all on". At that moment it becomes common knowledge that the Emperor is indeed naked. At that moment everyone knows that everyone knows that ... everyone knows that the Emperor is naked.

Let us consider how common knowledge among a group of agents can be formalised. For simplicity, we will denote the agents in the group by $0, 1, \ldots, n-1$, where n is the number of agents in the group. A sentence φ is then *common knowledge* of this group if the sentence

$$K_{i_1} K_{i_2} \cdots K_{i_m} \varphi \tag{3.4}$$

holds for all $m > 0$ and all choices of $i_1, i_2, \ldots, i_m < n$. The sentence (3.4) expresses that agent i_1 knows that agent i_2 knows that ... agent i_m knows that φ. In the traditional operator approach, it is impossible to express such infinitely many sentences as a single sentence. Therefore one is required to introduce extra symbols and axioms to take care of this modality. However, in the predicate approach we actually *can* express these infinitely many sentences in one single sentence. This is allowed through our ability to represent arbitrary recursive relations within the predicate theories. Let us show how.

Let $\mathsf{IterateK}$ be the following ternary relation over the natural numbers

$$\begin{aligned}\mathsf{IterateK} = \{(\ulcorner \varphi \urcorner, n, \ulcorner K_{i_1} K_{i_2} \cdots K_{i_m} \varphi \urcorner) \mid \varphi \text{ is a sentence, } m \geq 0, \\ \text{and } i_1, i_2, \ldots, i_m < n\}\,.\end{aligned}$$

The relation IterateK is easily seen to be recursive, so there is a formula

$$\mathit{IterateK}\,(x_1, x_2, x_3)$$

representing IterateK in any first-order agent theory. We can therefore express that (3.4) holds for all $m > 0$ and all $i_1, \ldots, i_m < n$ by the following *single* sentence in the agent language

$$\forall x \forall y \left(\mathit{IterateK}\,(\ulcorner\varphi\urcorner, n, x) \wedge y < n \rightarrow K_y\,(x)\right). \tag{3.5}$$

It is easy to show that all the sentences of the form (3.4) with $m > 0$ and $i_1, \ldots, i_m < n$ are logical consequences of (3.5). To see this, assume such m and $i_1, \ldots, i_m$ to be given. Then we obtain the following formal proof of (3.4) assuming (3.5)

1.	$\forall x \forall y \left(\mathit{IterateK}\,(\ulcorner\varphi\urcorner, n, x) \wedge y < n \rightarrow K_y\,(x)\right)$	(3.5)
2.	$\mathit{IterateK}\,(\ulcorner\varphi\urcorner, n, \ulcorner K_{i_2} \cdots K_{i_m}\urcorner) \wedge i_1 < n \rightarrow$ $K_{i_1}\,(\ulcorner K_{i_2} \cdots K_{i_m}\urcorner)$	instance of 1
3.	$\mathit{IterateK}\,(\ulcorner\varphi\urcorner, n, \ulcorner K_{i_2} \cdots K_{i_m}\urcorner)$	by def.
4.	$\mathit{IterateK}\,(\ulcorner\varphi\urcorner, n, \ulcorner K_{i_2} \cdots K_{i_m}\urcorner) \wedge i_1 < n$	3, $Q \vdash i_1 < n$
5.	$K_{i_1}\,(\ulcorner K_{i_2} \cdots K_{i_m}\urcorner)$	2, 4, MP

As usual, we abbreviate

$$K_{i_1}\,(\ulcorner K_{i_2} \cdots K_{i_m}\varphi\urcorner)$$

by

$$K_{i_1} K_{i_2} \cdots K_{i_m}\varphi,$$

so the derived sentence is indeed (3.4), as required.

What we have shown is that the representability of recursive relations gives us a simple way of formalising *common knowledge*. This is very important, since the common knowledge modality has been shown to be required in formalising many naturally occurring pieces of common sense reasoning. In particular, obtaining common knowledge has been shown to be a general prerequisite for coordinating actions and achieving agreements among groups of agents [Fagin *et al.*, 1995]. In the operator approach, common knowledge can only be introduced by extending the logical framework with new symbols and axioms to take care of this modality. As noted in Section 2.4.2, this seems to be an unsatisfactory solution [McCarthy, 1997]. We have shown that the predicate approach fares better in this respect, since common knowledge can be expressed purely in terms of the basic modality of knowing. From the example given here, it is not hard to see that many other modalities can be given a similar treatment using the representability of recursive relations.

Example 3.12. Let Neg be the following binary relation over the natural numbers

$$\mathsf{Neg} = \{(\ulcorner\varphi\urcorner, \ulcorner\neg\varphi\urcorner) \mid \varphi \text{ is a sentence of predicate logic}\}\,.$$

This is obviously a recursive relation, so there exists a formula $Neg\,(x_1, x_2)$ representing the relation in any first-order agent theory. The sentence

$$\forall x_1 \forall x_2\, (Neg\,(x_1, x_2) \rightarrow \neg\, (Kx_1 \wedge Kx_2))$$

therefore expresses that there is no φ such that both φ and $\neg\varphi$ are known. If an agent knew itself not to possess any contradictory knowledge, this could be expressed by the sentence

$$K\forall x_1 \forall x_2\, (Neg\,(x_1, x_2) \rightarrow \neg\, (Kx_1 \wedge Kx_2))\,.$$

The three examples above illustrate why we have requirement (i) in the definition of first-order agent theories (Definition 3.2): It ensures that all recursive relations become representable in any such theory, which allows the agent to express in a simple way many of the propositions we would like it to be able to express and reason about.

Remark. *Suppose M is a set of formulas of predicate logic. If $\varphi(x)$ represents the set of Gödel numbers of elements in M, we will often simply say that $\varphi(x)$ represents the set M. In this way formulas can be thought of as directly representing sets of other formulas.*

3.6. Consistency and Models

Questions of consistency and existence of various kinds of models play a central role in this thesis. In the following we will define these concepts. The notion of consistency used in the thesis is the standard one by which a first-order theory S is **consistent** if there is no formula φ such that both φ and $\neg\varphi$ are provable in S. The notion of an *interpretation* of S is also defined in the standard way. That is, an **interpretation** consists of a *domain* D together with a map I which maps each constant symbol c of S into an element c^I in D, each n-place function symbol f into a function $f^I : D^n \rightarrow D$ and each n-place predicate symbol P into a relation $P^I \subseteq D^n$. Truth in an interpretation is then defined in the usual way. A **model** of S is an interpretation in which every theorem of S is true. Consistency of a theory S is often proved by showing that the theory has a model.

In addition to the usual concept of an *interpretation* of a first-order theory, we will be using *Herbrand interpretations.* These are defined by the following.

Definition 3.13 (After Lloyd [1987])**.** *Let L be a language of first-order predicate logic. A* ***Herbrand interpretation*** *of L is an interpretation I of L satisfying the following requirements*

(i) *The domain of I is the set of closed terms (ground terms) of L.*
(ii) *I maps every constant symbol c of L into itself.*
(iii) *I maps every n-place function symbol f of L into the function that takes n closed terms $\tau_1, \dots, \tau_n$ and returns the closed term $f(\tau_1, \dots, \tau_n)$.*

Lemma 3.14 (After [Lloyd, 1987]). *Let L be a language of first-order predicate logic. Any Herbrand interpretation in L is uniquely determined by its set of true atomic sentences. That is, given a set of atomic sentences M in L, there is a unique Herbrand interpretation in which the set of true atomic sentences is M.*

Let S be a theory in a first-order language L and let I be an interpretation of L. If I is a Herbrand interpretation and furthermore a model of S we say that I is a **Herbrand model** of S. In mathematical logic, Herbrand models are sometimes referred to as *term models* or *closed term models*.

3.7. Reflection Principles

In this section we will be introducing the *reflection principles* that are needed to allow our agents to reason about their knowledge.

3.7.1. An Example. Consider again the example presented in Section 2.1. There we considered an agent with a knowledge base consisting of the following two sentences

$$K\, On\,(dotted, black) \tag{3.6}$$

$$K \forall x \forall y\, (On\,(x,y) \rightarrow \neg On\,(y,x))\,. \tag{3.7}$$

This knowledge base becomes a first-order agent theory if we add the axioms of Q. Let us refer to the thus obtained first-order agent theory as S. Since S is a theory of predicate logic, it has the two inference rules *modus ponens* (MP) and *generalisation* (Gen). These are inference rules that the agent can use to deduce new facts from the facts represented as axioms in S. As mentioned in Section 2.1, we would like the agent to be able to deduce the sentence

$$K \neg On\,(black, dotted)\,. \tag{3.8}$$

from the two axioms (3.6) and (3.7). However, this sentence is not provable in S, so the agent will not be able to infer that it holds. The problem is that there are no axioms or inference rules in S allowing the agent to reason with its own knowledge. For the agent to be of any use, we need to provide it with such axioms and inference rules.

Suppose we extend S to a theory S' by adding the following axiom schemes

$$K\,(\varphi \rightarrow \psi) \rightarrow (K\varphi \rightarrow K\psi)\,, \text{ for all sentences } \varphi \text{ and } \psi. \tag{3.9}$$

$$K\varphi, \text{ for all theorems } \varphi \text{ of predicate logic.} \tag{3.10}$$

The axiom scheme (3.9) expresses that implicit knowledge is closed under modus ponens. Concerning explicit knowledge, the axiom scheme gives us that if the agent explicitly knows $\varphi \to \psi$ and explicitly knows φ, then it can in one inference step come to explicitly know ψ. The axiom scheme (3.10) expresses that the agent (explicitly) knows predicate logic. Adding these two axiom schemes to the knowledge base is sufficient to allow the agent to derive a number of desirable consequences of its own knowledge. For instance we have the following proof in S'

1.	$K \forall x \forall y \, (On\,(x,y) \to \neg On\,(y,x))$	axiom (3.7)
2.	$K \big(\forall x \forall y \, (On\,(x,y) \to \neg On\,(y,x)) \to$ $(On\,(dotted, black) \to \neg On\,(black, dotted))\big)$	instance of (3.10)
3.	$K\,(On\,(dotted, black) \to \neg On\,(black, dotted))$	using inst. of (3.9) on 1, 2
4.	$K\,On\,(dotted, black)$	axiom (3.6)
5.	$K \neg On\,(black, dotted)$	using inst. of (3.9) on 3, 4

This proof shows that the agent is able to come to the knowledge that the black block is not on the dotted block from the knowledge that the dotted block is on the black block (expressed by axiom (3.6)) and the knowledge that the *On*-relation is asymmetric (expressed by axiom (3.7)).

The formal proof leading to the conclusion $K \neg On\,(black, dotted)$ involves instances of the axiom schemes (3.9) and (3.10). These are very general axiom schemes that should be included in the knowledge bases of agents to allow them to reason about their own knowledge and infer consequences of it. As another example of the use of these axiom schemes, we can note that in the context of Example 3.10, these axioms would allow the agent to deduce

$$K\,(\neg K\,On\,(striped, dotted) \wedge \neg K \neg On\,(striped, dotted))$$

from

$$K \forall x\,(About\,(x, striped) \to \neg K\,x)\,.$$

Axiom schemes such as (3.9) and (3.10) which express general principles by which an agent can infer consequences of its knowledge we choose to call *reflection principles*. In the literature they are often referred to as *epistemic principles*, but since axiom schemes such as (3.9) and (3.10) are also consistent with interpreting K as denoting for instance *belief*, *provability* or *truth*, we choose the more neutral term *reflection principle*. We should note, however, that the term *reflection principle* is often used in a much more narrow sense in mathematical logic (introduced by Feferman [1962]).

If the intended interpretation of K is belief rather than knowledge, then the axiom schemes (3.9) and (3.10) express that (implicit) belief is closed under modus ponens and that every theorem of predicate logic is believed. These might be considered as reasonable principles for belief to satisfy as well. If the intended interpretation of $K\varphi$ is "φ is provable in S" for some first-order theory S or "φ is true in I" for some interpretation of predicate logic I, then it is seen that the axiom schemes (3.9) and (3.10) express valid principles regarding both of these intended interpretations (both provability and truth are closed under modus ponens, and every theorem of predicate logic is both provable and true). Interpreting reflection principles as concerning *knowledge* or *belief* is traditionally conceived as being the subjects of *epistemic logic* and *doxastic logic*, respectively. Interpreting them as concerning *provability* or *truth* are the subjects of *provability logic* and *theories of truth*, respectively. The consistency results for reflection principles that we are going to prove in this thesis can be interpreted as concerning any of these four subjects (epistemic logic, doxastic logic, provability logic, theories of truth). All four subjects are concerned with *reflection* in the general sense of *representing aspects of an object within this object itself*: In the two former subjects, an agent's knowledge (belief) about its own knowledge base is represented within this knowledge base itself; in the two latter subjects, provability (truth) within a formal system is represented within this formal system itself.

3.7.2. Definition of Reflection Principles. In this thesis, we will restrict our attention to the following familiar set of reflection principles.

Definition 3.15. *By a* ***reflection principle*** *we understand any of the following axiom schemes in first-order predicate logic.*

A1. $K\varphi \to \varphi$.
A2. $K(K\varphi \to \varphi)$.
A3. $K\varphi$, *if* φ *is a theorem in* Q.
A4. $K(\varphi \to \psi) \to (K\varphi \to K\psi)$.
A5. $\neg(K\varphi \wedge K\neg\varphi)$.
A6. $K\varphi \to KK\varphi$.
A7. $\neg K\varphi \to K\neg K\varphi$.
T. $K\varphi \leftrightarrow \varphi$.

These reflection principles are *axiom schemes*. Unless otherwise stated we will consider them to be ranging over *all* sentences φ and ψ of predicate logic. Thus for instance reflection principle A1 stands for the infinite set of sentences given by

$$\{K\varphi \to \varphi \mid \varphi \text{ is a sentence of predicate logic}\}.$$

When we want to consider a reflection principle as ranging only over a restricted set of sentences M, we will refer to it as the reflection principle *instantiated over* M. Thus for instance the reflection principle A1 instantiated over M will be the following set of sentences

$$\{K\varphi \to \varphi \mid \varphi \in M\}.$$

Instances of a reflection principle over a set M are called *M-instances* of the reflection principle.

Note that the reflection principles can be seen as simple translations into predicate logic of standard axioms within propositional modal logic. Reflection principle A1 is the translation of the modal axiom **T**: $\Box p \to p$; A4 is the translation of the modal axiom **K**: $\Box(p \to q) \to (\Box p \to \Box q)$; A5 is the translation of **D**: $\neg(\Box p \wedge \Box\neg p)$; A6 is the translation of **4**: $\Box p \to \Box\Box p$; A7 is the translation of **5**: $\neg\Box p \to \Box\neg\Box p$; and T is the translation of **Triv**: $\Box p \leftrightarrow p$ (the names **T**, **K**, **D**, **4**, **5** and **Triv** are taken from Hughes and Cresswell [1996]).[3] The details of this translation from modal logic formulas into predicate logic can be found in des Rivières and Levesque [1988]. The reader familiar with such translations should note that the translation used here does does not correspond to the standard embedding of propositional modal logic into predicate logic, where one encodes the possible-worlds semantics in the translation [Andréka *et al.*, 1995].

3.7.3. Interpretation of the Reflection Principles. Let us suppose that the predicate symbol K is intended to express the knowledge of an agent. Then the various reflection principles can be interpreted in the following way. Reflection principle A1 is expressing that everything known by the agent is true—in other words, that only true things are known. Reflection principle A2 expresses that the agent knows A1, that is, the agent knows that only true things are known. Reflection principle A3 expresses that arithmetic is known, and A4 that (implicit) knowledge is closed under modus ponens. Reflection principle A5 is expressing that the agent has no contradictory knowledge. Reflection principles A6 and A7 are known as the principles of *positive introspection* and *negative introspection*, respectively [Fagin *et al.*, 1995]. If A6 is an axiom scheme in the knowledge base of the agent, and if the agent knows φ, then it will be able to use A6 to infer that *it knows that it knows* φ. Reflection principle A6 thus expresses the agent's introspective awareness of its own knowledge ("it knows what it knows" [Fagin *et al.*, 1995]). Conversely, reflection principle A7 expresses the agent's introspective awareness of its own non-knowledge.

All of the axiom schemes A1–A7 seem to express natural principles concerning knowledge. If we interpret K as expressing belief rather than knowledge, then A1 might no longer be a realistic principle, since an agent might have incorrect beliefs. However, axiom schemes A2–A7 all seem to be consistent with interpreting K as belief.

[3]Note that we use T to denote the principle $K\varphi \leftrightarrow \varphi$ rather than the principle corresponding to the modal axiom T.

We have not yet mentioned reflection principle T, which is the strongest of them all. It is usually only applied when K is intended to express *truth*. It is recognised as Tarski's famous **schema T**, which Tarski presented as the *material adequacy principle* for K to be interpreted as a *truth predicate* [Tarski, 1944; 1956]. The idea is that if $K\varphi$ expresses that "φ is true", then schema T expresses the principle that

$$\text{“}\varphi \text{ is true” if and only if } \varphi,$$

which is the fundamental principle characterising the predicate "is true".

In addition to our interest in schema T as a principle of truth, there is a more technical reason for including it among the reflection principles. This is that schema T logically entails all of the other reflection principles. More precisely, we have the following result.

Lemma 3.16. *Let L be a first-order agent language, and let M be a set of sentences in L satisfying the following closure property:*

$$\textit{If } \varphi, \psi \in M \textit{ then } K\varphi, \neg\varphi, \varphi \wedge \psi \in M.$$

Let S be an agent theory in L in which all M-instances of reflection principle T are theorems. Then all M-instances of reflection principles A1–A7 are theorems in S as well.

Proof. Let φ and ψ be arbitrary sentences in M. We have to prove that A1–A7 are theorems in S when instantiated with these sentences. By the closure property of M, all of the following are sentences in M

$$\neg\varphi, \varphi \rightarrow \psi, K\varphi, \neg K\varphi, K\varphi \rightarrow \varphi. \tag{3.11}$$

The second sentence is an abbreviation of $\neg(\varphi \wedge \neg\psi)$ and the last sentence is an abbreviation of $\neg(K\varphi \wedge \neg\varphi)$. We now have the following formal proof in S

1.	$K\varphi \leftrightarrow \varphi$	M-instance of T
2.	$K\varphi \rightarrow \varphi$	1, biconditional elimination
3.	$K(K\varphi \rightarrow \varphi) \leftrightarrow (K\varphi \rightarrow \varphi)$	M-instance of T
4.	$(K\varphi \rightarrow \varphi) \rightarrow K(K\varphi \rightarrow \varphi)$	3, biconditional elimination
5.	$K(K\varphi \rightarrow \varphi)$	2, 4, MP

In this proof, line 2 is A1 and line 5 is A2. Thus these must be theorems of S. A3–A7 are shown to be theorems of S in a similar manner using instances of T over the sentences in (3.11). We leave the details to the reader. □

As shown by the example given in the beginning of this section, reflection principles as the ones we have been presenting above can be used by an agent to infer important consequences of its own knowledge (belief). We would therefore suggest to include a subset of these principles in any first-order agent theory. Unfortunately, it turns out that if we take a first-order agent theory and add such a subset of reflection principles as axioms, then the theory is most likely to become inconsistent. The problem is that the reflection principles will give the agent sufficient reasoning powers to be able to reason about self-referential sentences and thereby engage in paradoxical reasoning such as involved in the knower paradox (Sections 1.4.1 and 1.4.2).

We will present the inconsistency results related to the reflection principles in detail in the following chapter. After that we will proceed to present our graph theoretical approach to circumventing these inconsistencies. These inconsistencies are the main threat to the construction of logical frameworks for introspective agents in the predicate approach. Finding ways to circumvent them could thus reasonably be conceived as the most important task in the predicate approach to formalising the knowledge and belief of agents.

3.8. Chapter Notes

The introduction of first-order predicate logic in Section 3.1 is adapted from Mendelson [1997]. There are a number of deviations from his presentation, however. These are deviations due to the author with the purpose of making a framework which is simpler to deal with in connection with formalising knowledge and belief. One of these deviations is the non-standard definition of Q. Even with our deviations, we stay completely within classical first-order predicate logic. The definition of first-order agent languages and theories in Section 3.2 is due to the author. These are simply first-order languages and theories satisfying a few extra requirements. In principle, we could just as well have implicitly required *all* languages and theories of first-order predicate logic to satisfy these requirements. The examples in Section 3.5.2 are due to the author as well. The author is not aware of similar approaches to formalising the modalities of *knowing about* and *common knowledge*. Such modalities are usually not introduced through arithmetical formulas representing recursive relations, but rather through extending the language with new operators or predicates and corresponding axioms. We argued against doing things in this way in Section 2.4.2. Our examples show that an agent do not have to be equipped with a tailormade language in order for it to be able to express and reason about such modalities.

Formalising knowledge as a predicate within a language of first-order predicate logic is a well-known approach. It can be found e.g. in des Rivières and Levesque [1988], Morreau and Kraus [1998] and Grant *et al.* [2000]. The two first of these articles also introduce the reflection principles A1–A7 given in Section 3.7.2. They do not refer to them as *reflection principles*, however, but simply as *axioms*. Lemma 3.16 appears in Bolander [2002a].

CHAPTER 4

Problems of Introspection and Self-Reference

In this chapter we will show in which way self-reference becomes a problem in the construction of strongly introspective agents. More precisely, we will show that equipping agents with a number of reflection principles will in many cases allow them to reason about self-referential sentences in a paradoxical way. This implies that the knowledge bases of the agents become *inconsistent*. In this chapter we present the well-known inconsistency results by Montague and Thomason concerning this problem. We also present an original inconsistency result related to the problem of constructing agents with *perfect introspection*.

After having proved these results, we turn to the problem of how the inconsistencies can be avoided, that is, how we can ensure that our agents will not engage in paradoxical reasoning. We discuss different approaches to this problem, and present the particular approach forming the basis of this thesis. Two known results within this approach are presented: the Rivières-Levesque theorem and the Morreau-Kraus theorem.

We start the chapter by giving a motivating example illustrating how self-reference can unexpectedly cause the inconsistency of a knowledge base.

Example 4.1 (The *About* predicate)**.** This example is the promised continuation of Example 3.10. We again assume that we are given a formula $About(x_1, x_2)$ representing the relation About given by

$$\mathsf{About} = \{(\ulcorner\varphi\urcorner, \ulcorner c\urcorner) \mid \varphi \text{ is a sentence and } c \text{ is a constant symbol occurring in } \varphi\}.$$

An agent will then be able to express that it has no knowledge about a striped block by

$$K\forall x\,(About(x, striped) \to \neg Kx).$$

However, does this really work? Suppose S is a first-order agent theory constituting the knowledge base of the agent in question. Suppose further that S contains the reflection principle A1 ("Everything known is true"). If the agent knows that it does not have any knowledge about a striped block, we should have

$$S \vdash K\forall x\,(About(x, striped) \to \neg Kx). \tag{4.1}$$

Let φ denote the sentence $\forall x\,(About\,(x, striped) \rightarrow \neg Kx)$. Then (4.1) can be abbreviated as

$$S \vdash K\varphi. \tag{4.2}$$

Note that by definition of $About\,(x_1, x_2)$, we have

$$S \vdash About\,(\varphi, striped)\,. \tag{4.3}$$

Now the agent will be able to carry out the piece of reasoning given by the following formal proof in S

1.	$K\varphi$	(4.2)
2.	$K\varphi \rightarrow \varphi$	instance of A1
3.	φ	1, 2, MP
4.	$\forall x\,(About\,(x, striped) \rightarrow \neg Kx)$	same as 3
5.	$About\,(\varphi, striped) \rightarrow \neg K\varphi$	instance of 4
6.	$About\,(\varphi, striped)$	(4.3)
7.	$\neg K\varphi$	5, 6, MP
8.	$K\varphi \wedge \neg K\varphi$	1, 7, conj. introduction

We have hereby derived a contradiction in S, so S is inconsistent! How could this happen? At first it does not seem that we have any particularly problematic axioms in S that could cause this inconsistency. The only non-logical axioms we are using in the proof above are the ones in Q (which give us line 6), one instance of A1 (line 2), and the sentence expressing that the agent knows that nothing is known about the striped block (line 1). However, it turns out that problems of self-reference are lurking in the background.

The sentence $K\varphi$ is actually self-referential. It is so for the following reason. It expresses that the agent knows φ, where φ expresses that nothing is known about the striped block. But if the agent knows φ then it actually *does* know something about the striped block: it knows that it knows nothing about it. So $K\varphi$ is actually a piece of knowledge contradicting itself: it expresses the non-knowledge of a set of sentences (those concerning the striped block) of which it is itself an element. The inconsistency derived from this sentence is structurally very closely related to the liar paradox in the original version in which a Cretan is stating: "All Cretans are liars".

A case of self-referential knowledge similar to the above would be the following: "I know only two things about Malaysia". Then it seems that unless we are very careful in the way we interpret the word "about", this very statement would constitute a third piece of knowledge about Malaysia.

The problem of self-reference occurring in these examples can be avoided by giving an alternative definition of the *About* predicate. We will do this in Example 6.10. At this point we just want to draw the readers attention to the following facts illustrated by the example:

1) Self-reference can occur in sentences that at first do not seem to be self-referential, or at least are not constructed with the purpose of being self-referential. Such a sentence is $K\varphi$ above, which was constructed entirely with the purpose of expressing an agents knowledge of its own ignorance concerning the properties of a certain object.

2) Reasoning about such self-referential sentences can be paradoxical and thereby cause the theories in which the reasoning takes place to become inconsistent. In the case of first-order agent theories this is fatal, since then everything becomes provable in the theory. If the theory is the knowledge base of an agent, it means that the agent will become useless.

In the following we will give a number of general theorems showing how self-referential reasoning can cause inconsistency. We will then look at the possibilities of regaining consistency by suitably restricting this kind of reasoning.

4.1. The Diagonalisation Lemma

We have already mentioned that the predicate approach to knowledge and belief allows agents to form and reason about self-referential statements. In this section we will give the details of how this is possible, in that we will give a general method by which such sentences can be constructed.

Self-reference and its properties is an interesting subject in itself. Many formal languages have been developed which are explicitly equipped with the means to form self-referential statements. One such language is proposed by Smullyan [1984]. In his system, there is a *chameleonic term* σ which is always interpreted as denoting the formula in which it itself occurs. It is thus a term with a context dependent semantics, and it plays more or less the same role in the formal language as the term "this sentence" does in natural language. In such a formal language there is a very straightforward way to formalise sentences such as "this sentence is not true" (the liar sentence) or "this sentence is not known by Mr. X" (the knower sentence). A related approach to formalising self-referential statements is taken by Barwise and Etchemendy [1987], where the formal language is equipped with a term "this" which is similarly intended to denote the sentence in which it occurs.

First-order predicate logic does not have any such direct means to formulate self-referential statements. There is, however, an *indirect* method by which one can construct self-referential sentences in predicate logic (or, at least, sentences that *behave* as self-referential ones). This method is known as the *diagonalisation method*. It is named so because it is inspired by—and structurally closely related to—the diagonalisation method used originally by

Cantor to prove the uncountability of the powerset of the set of natural numbers [Cantor, 1891]. The diagonalisation method in logic is most often presented in the form of the so-called *diagonalisation lemma*. We state and prove a version of this lemma below. The lemma involves a *diagonalisation function* $d : \mathbb{N} \to \mathbb{N}$ defined by

$$d(\ulcorner\varphi(x)\urcorner) = \ulcorner\varphi(\ulcorner\varphi(x)\urcorner)\urcorner, \text{ for all formulas } \varphi(x) \text{ with exactly } x \text{ free.}^1$$

With the usual abbreviations, this equality can be written as $d(\ulcorner\varphi(x)\urcorner) = \ulcorner\varphi(\varphi(x))\urcorner$. The function d is recursive, so by Theorem 3.9 there exists a formula $D(x_1, x_2)$ representing d in Q. Given the formula $D(x_1, x_2)$, the diagonalisation lemma can be formulated in the following way.

Lemma 4.2 (The diagonalisation lemma. Mendelson [1997]). *Let $\gamma(x)$ be a formula of predicate logic with only x free. Define formulas $\alpha(x_1)$ and β by $\alpha(x_1) = \forall x_2\,(D(x_1, x_2) \to \gamma(x_2))$ and $\beta = \forall x_2\,(D(\alpha, x_2) \to \gamma(x_2)) = \alpha(\alpha)$. Then*

$$Q \vdash \beta \leftrightarrow \gamma(\beta).$$

Proof. The implication from left to right is proved by the following formal proof.

1.	β	assumption
2.	$\forall x_2\,(D(\alpha, x_2) \to \gamma(x_2))$	same as 1
3.	$D(\alpha, \beta) \to \gamma(\beta)$	instance of 2
4.	$D(\alpha, \beta)$	$D(x_1, x_2)$ represents d (in Q)
5.	$\gamma(\beta)$	3, 4, MP
6.	$\mathrm{Q}, \beta \vdash \gamma(\beta)$	1–5

From line 6 we now immediately get $Q \vdash \beta \to \gamma(\beta)$ by the deduction theorem. This proves the implication from left to right. The opposite implication is proved by the following.

1.	$\gamma(\beta)$	assumption
2.	$D(\alpha, x_2)$	assumption
3.	$\exists! x_2 D(\alpha, x_2)$	$D(x_1, x_2)$ represents d
4.	$\exists x_2 D(\alpha, x_2) \land$ $\forall x_2 \forall x_3\,(D(\alpha, x_2) \land D(\alpha, x_3) \to x_2 = x_3)$	same as 3
5.	$\forall x_2 \forall x_3\,(D(\alpha, x_2) \land D(\alpha, x_3) \to x_2 = x_3)$	4, conj. elimination
6.	$D(\alpha, x_2) \land D(\alpha, \beta) \to x_2 = \beta$	instance of 5

[1] This condition actually only gives a partial function on $\mathbb{N}$, since d only assigns values to arguments that are Gödel numbers of particular sentences. We will here and throughout the thesis simply assume that such partially defined functions are assigned the value 0 for all other arguments.

7.	$D(\alpha, \beta)$	$D(x_1, x_2)$ represents d
8.	$D(\alpha, x_2) \rightarrow x_2 = \beta$	by 6, 7
9.	$x_2 = \beta$	2, 8, MP
10.	$x_2 = \beta \rightarrow (\gamma(\beta) \rightarrow \gamma(x_2))$	instance of P7
11.	$\gamma(\beta) \rightarrow \gamma(x_2)$	9, 10, MP
12.	$\gamma(x_2)$	1, 11, MP
13.	$Q, \gamma(\beta), D(\alpha, x_2) \vdash \gamma(x_2)$	1–12
14.	$Q, \gamma(\beta) \vdash D(\alpha, x_2) \rightarrow \gamma(x_2)$	13, deduction theorem
15.	$Q, \gamma(\beta) \vdash \forall x_2 \, (D(\alpha, x_2) \rightarrow \gamma(x_2))$	14, Gen
16.	$Q, \gamma(\beta) \vdash \beta$	same as 15

From line 16 we immediately get $Q \vdash \gamma(\beta) \rightarrow \beta$. The proof is therefore complete. $\square$

The proof above relies on an instance of axiom scheme P7 (p. 48), which is one of the axioms of equality. Without this axiom the proof would not go through. The diagonalisation lemma is therefore only a result that holds in predicate logic *with equality* or, alternatively, only for *theories with equality* (a *theory with equality* is a theory that has P6 and all the relevant instances of P7 as theorems).

It is probably not immediately obvious in which way the diagonalisation lemma is related to self-reference. We will try to explain this now. In general, self-referential sentences can be put on the following form

$$S : \text{Sentence } S \text{ has the property } P. \tag{4.4}$$

Here S is considered to be a *name* referring to the sentence "Sentence S has the property P". We already saw two examples of sentences given on this form in Section 1.4.1: the *liar sentence* and the *knower sentence* given by, respectively,

$$L : \text{Sentence } L \text{ is not true.}$$

$$S : \text{Sentence } S \text{ is not known by Mr. X.}$$

Alternatively, one could use the demonstrative "this sentence" to obtain self-reference. In that case, a sentence of the form (4.4) would be expressed as: "This sentence has the property P". The liar sentence would then become "This sentence is not true" and the knower sentence would become "This sentence is not known by Mr. X".

Any sentence S given on the form (4.4) must satisfy the following *semantic principle*

$$S \text{ is true if and only if "Sentence } S \text{ has the property } P\text{" is true.} \tag{4.5}$$

This semantic principle holds simply because S is a name referring to the sentence "Sentence S has the property P". It is this semantic principle we apply whenever we reason about self-referential sentences. Let us try to compare the principle to the diagonalisation lemma. In the diagonalisation lemma a sentence β is constructed which satisfies

$$Q \vdash \beta \leftrightarrow \gamma(\beta). \tag{4.6}$$

We can think of the subformula $\gamma(\beta)$ as saying "β has the property expressed by γ". We can therefore read the equivalence in (4.6) as expressing

$$\beta \text{ is true if and only if "}\beta \text{ has the property expressed by } \gamma\text{" is true.} \tag{4.7}$$

But this is the same as (4.5) when replacing S by β and replacing "the property P" by "the property expressed by γ". Thus the equivalence (4.6) proved by the diagonalisation lemma is actually expressing the *semantic principle* which characterises a self-referential sentence β given by

$$\beta : \text{Sentence } \beta \text{ has the property expressed by } \gamma. \tag{4.8}$$

Thus we can think of the sentence β constructed in the diagonalisation lemma as being a self-referential sentence saying of itself that it has the property expressed by γ. This means that we can interpret the diagonalisation lemma as proving that given any property expressible by a formula γ, there is a sentence β saying of itself that it has this property (or, at least, it behaves semantically as if it were such a sentence). If γ is the formula $\neg Kx$, then β says of itself that it is not known. This is a formal variant of the knower sentence. We will consider such formal knower sentences many times in the following, so they deserve to be given a separate definition.

Definition 4.3. *By a* ***formal knower sentence*** *we understand a sentence β satisfying*

$$Q \vdash \beta \leftrightarrow \neg K\beta.$$

The diagonalisation lemma guarantees that such a sentence exists. Note that a formal knower sentence is the same as a *formal liar sentence*: If we interpret K as expressing truth rather than knowledge, then the sentence β will be expressing that it itself is untrue.

4.2. The Inconsistency Results

We have chosen the predicate approach to formalising the knowledge and belief of agents, since it provides the agents with a much higher expressive power than the traditional operator approach. For instance it allows the agents to express modalities such as *common knowledge* and *knowing more* (Examples 3.11 and 2.1) in a simple way. It also allows the agents to express strongly introspective knowledge such as "I know that I do not have any contradictory knowledge": $K\forall x_1 \forall x_2 \left(Neg\left(x_1, x_2\right) \rightarrow \neg \left(Kx_1 \wedge Kx_2\right)\right)$ (Example 3.12). What we have been showing in the previous section is that in addition to all of this, the predicate approach allows the agents to form self-referential sentences such as: "This sentence is not known" (the knower sentence). This is permitted through the diagonalisation lemma. Unfortunately, self-referential sentences are known to be potential troublemakers. As many of these sentences are denying their own properties, they seem to be in conflict with some of our most basic semantic understandings and with the way we formally treat such semantics. Self-referential sentences concerning knowledge is no exception to this fact.

In the following we will show a number of results to the effect that self-referential sentences concerning knowledge, belief and truth can cause the inconsistency of the logical frameworks in which they are formulated. These inconsistencies only appear in logical theories that have been equipped with a number of *reflection principles*, since the reflection principles are what allows us to reason about the relevant sentences (see Section 3.7). The diagonalisation lemma is at the heart of all our inconsistency results below. More precisely, all results below are based on reasoning about a formal knower sentence, and such a sentence is guaranteed to exist by the diagonalisation lemma.

4.2.1. Tarski's Theorem. The first inconsistency result is a version of Tarski's famous theorem on the undefinability of truth.

THEOREM 4.4 (Tarski's theorem. After Tarski [1956]). *The theory Q extended with reflection principle T (schema T) is **inconsistent**.*

PROOF. Recall that reflection principle T is the axiom scheme given by

$$K\varphi \leftrightarrow \varphi.$$

Let β be a formal knower sentence, that is, a sentence satisfying

$$Q \vdash \beta \leftrightarrow \neg K\beta.$$

Since we have

$$T \vdash K\beta \leftrightarrow \beta,$$

we immediately get

$$Q, T \vdash K\beta \leftrightarrow \neg K\beta.$$

This shows that the theory $Q \cup T$ is inconsistent. □

Since schema T expresses the principle that K must satisfy in order to be interpreted as a *truth predicate* (Section 3.7.3), Tarski's theorem above can be seen as a proof of the impossibility of having a (formal) language containing a truth predicate for that same language [Tarski, 1956]. If we intend to interpret K as a *knowledge predicate* rather than a *truth predicate*, then schema T expresses that everything known is true (this is the implication $K\varphi \rightarrow \varphi$, which is identical to reflection principle A1) and that everything true is known (this is the implication $\varphi \rightarrow K\varphi$). That is, schema T expresses a principle of *correct* and *complete* knowledge. Tarski's theorem shows that having both correct and complete knowledge is unattainable in the predicate approach.

The apparent impossibility of having an agent with both *correct* and *complete* knowledge has not been a great worry to researchers in artificial intelligence. It does not seem to be a realistic property of an agent to have anyway. The following two inconsistency results have been met with much more concern, however, since they prove that an agent can not even consistently be equipped with reflection principles expressing quite basic and natural properties of knowledge and belief.

4.2.2. Montague's Theorem. The following theorem can be interpreted as showing that we can not give a consistent treatment of *knowledge* using the predicate approach.

Theorem 4.5 (Montague's theorem. After Montague [1963]). *The theory Q extended with reflection principles A1–A4 is* ***inconsistent****.*

Proof. Let β be a formal knower sentence. Then we have the following proof in Q extended with A1–A4

1.	$K(\beta \rightarrow \neg K\beta)$	A3, since $Q \vdash \beta \leftrightarrow \neg K\beta$
2.	$K(K\beta \rightarrow \beta)$	A2
3.	$K\big((K\beta \rightarrow \beta) \rightarrow ((\beta \rightarrow \neg K\beta) \rightarrow \neg K\beta)\big)$	A3 on tautology
4.	$K\big((\beta \rightarrow \neg K\beta) \rightarrow \neg K\beta\big)$	2, 3, A4
5.	$K\neg K\beta$	1, 4, A4
6.	$K(\neg K\beta \rightarrow \beta)$	A3, since $Q \vdash \beta \leftrightarrow \neg K\beta$
7.	$K\beta$	5, 6, A4
8.	$\beta \rightarrow \neg K\beta$	since $Q \vdash \beta \leftrightarrow \neg K\beta$
9.	$K\beta \rightarrow \beta$	A1
10.	$(K\beta \rightarrow \beta) \rightarrow ((\beta \rightarrow \neg K\beta) \rightarrow \neg K\beta)$	tautology
11.	$(\beta \rightarrow \neg K\beta) \rightarrow \neg K\beta$	9, 10, MP
12.	$\neg K\beta$	8, 11, MP
13.	$K\beta \wedge \neg K\beta$	7, 12, conj. introduction

This proof shows that Q extended with A1–A4 is inconsistent, since the contradiction $K\beta \wedge \neg K\beta$ is provable within that theory. □

A few things should be noted about the proof above. The reader familiar with Montague's proof will notice that our proof is considerably simpler and involves a much simpler self-referential sentence. We have been able to give this simpler proof because our reflection principle A3 is slightly stronger than the corresponding axiom scheme considered by Montague. Instead of the reflection principle A3, Montague has

A3′. $K\varphi$, if φ is a logical axiom of predicate logic or an axiom of Q.

Note that in the proof above, the lines 1–5 form a precise "modalised copy" of the lines 8–12: each formula φ appearing in lines 8–12 appear as $K\varphi$ in the lines 1–5. This means that in the proof we carry out the same piece of reasoning at two different levels: the "objective level" and the "knowledge level". The proof is actually nothing more than a formalisation of the knower paradox, which we presented in a slightly less formal way in Section 1.4.2. What we have obtained by the above theorem is therefore to show that if an agent has the reflection principles A1–A4 in its knowledge base, then it will be able to carry out the reasoning involved in the knower paradox, and the knowledge base will therefore become inconsistent.

Since all of the reflection principles A1–A4 express natural principles of knowledge (cf. Section 3.7.3), Montague's theorem apparently prove that knowledge can not be treated consistently in the predicate approach. To preserve the predicate approach we therefore need to circumvent this inconsistency result somehow. The main theme of this thesis is to find ways of doing this.

4.2.3. Thomason's Theorem. The following theorem shows that even if we exclude both of the strong reflection principles T and A1, we can still find an inconsistent combination of the principles.

Theorem 4.6 (Thomason's theorem. After Thomason [1980])**.** *The theory Q extended with reflection principles A2–A6 is* ***inconsistent****.*

Proof. Let β be a formal knower sentence. Note that the first 8 lines of the proof of Montague's theorem above are carried out in the theory Q extended with A2–A4. We therefore already know from lines 5 and 7 in this proof that

$$\mathrm{Q}, \mathrm{A2–A4} \vdash K\neg K\beta \tag{4.9}$$

and

$$\mathrm{Q}, \mathrm{A2–A4} \vdash K\beta. \tag{4.10}$$

We now have the following proof in Q extended with A2–A6

1.	$K\neg K\beta$	(4.9)
2.	$K\beta$	(4.10)
3.	$K\beta \to KK\beta$	A6
4.	$KK\beta$	2, 3, MP
5.	$KK\beta \wedge K\neg K\beta$	1, 4, conj. introduction
6.	$\neg(KK\beta \wedge K\neg K\beta)$	A5
7.	$(KK\beta \wedge K\neg K\beta) \wedge \neg(KK\beta \wedge K\neg K\beta)$	5, 6, conj. introduction

Line 7 is a contradiction, so the theory in question must be inconsistent. □

Montague's theorem from the previous section proved that we can not consistently get a formalisation of knowledge satisfying all the principles we would like knowledge to satisfy. Thomason's theorem can be considered to give a similar conclusion for the case in which K is interpreted to denote *belief* rather than *knowledge*. In Section 3.7.3 we noted that all the reflection principles A2–A7 express natural properties of belief, and the theory shown to be inconsistent by Thomason's theorem only contains reflection principles from this group.

4.2.4. Inconsistency of Perfect Introspection. We now give our final inconsistency result. It is inspired by the theorems of Montague and Thomason.

Theorem 4.7. *The theory Q extended with reflection principles A3–A7 is **inconsistent**.*

Proof. Let β be a formal knower sentence. Then we have the following proof in Q extended with A3–A7

1.	$K(\beta \to \neg K\beta)$	A3, since $Q \vdash \beta \leftrightarrow \neg K\beta$
2.	$K(\beta \to \neg K\beta) \to (K\beta \to K\neg K\beta)$	A4
3.	$K\beta \to K\neg K\beta$	1, 2, MP
4.	$K\beta \to KK\beta$	A6
5.	$K\beta \to KK\beta \wedge K\neg K\beta$	from 3, 4
6.	$\neg(KK\beta \wedge K\neg K\beta) \to \neg K\beta$	5, contraposition
7.	$\neg(KK\beta \wedge K\neg K\beta)$	A5
8.	$\neg K\beta$	6, 7, MP
9.	$\neg K\beta \to K\neg K\beta$	A7
10.	$K\neg K\beta$	8, 9, MP
11.	$K(\neg K\beta \to \beta)$	A3, since $Q \vdash \beta \leftrightarrow \neg K\beta$
12.	$K(\neg K\beta \to \beta) \to (K\neg K\beta \to K\beta)$	A4
13.	$K\neg K\beta \to K\beta$	11, 12, MP
14.	$K\beta$	10, 13, MP

15. $K\beta \wedge \neg K\beta$ 8, 14, conj. introduction

This shows that Q extended with A3–A7 is inconsistent. □

Let us try to interpret this theorem. Suppose that Q and A3–A7 are the axioms in the knowledge base of an agent (these axioms constitute a first-order agent theory). Suppose further that K is intended to denote *belief*. We have not included reflection principles A1 and A2 in the knowledge base, so there is neither a claim that beliefs should be correct, nor a claim that the agent believes its own beliefs to be correct. What we have is the following

(i) Arithmetic is believed (reflection principle A3).
(ii) No contradictions are believed (reflection principle A4).
(iii) Implicit belief is closed under modus ponens (reflection principle A5).
(iv) If something is believed, then the agent can in one inference step come to believe that it is believed (reflection principle A6).
(v) If something is *not* believed, then the agent can in one inference step come to believe that it is not believed (reflection principle A7).

When we have reflection principles A6 and A7 together, as we do in this case, we say that the agent has **perfect introspection** [Lakemeyer, 1992]. The properties expressed by A6 and A7 are given as (iv) and (v) above. Perfect introspection expresses that the agent has full introspective awareness of its own beliefs: whether something is believed or not, the agent will always (implicitly) know which one is the case. Lakemeyer puts it this way: "Agents with perfect introspection may have incomplete beliefs about the world, but they possess complete knowledge about their own beliefs" [Lakemeyer, 1992]. The theorem above shows that there can not exist an agent whose beliefs satisfy all of the properties (i)–(v), since any agent theory containing reflection principles A3–A7 will be inconsistent. Thus we apparently also have to give up the goal of having agents with *perfect introspection* if we want to use the predicate approach in formalising the beliefs of these agents.

4.2.5. Concluding Remarks. The four theorems above are collected into the following corollary.

Corollary 4.8. *The theory Q extended with any of the following sets of reflection principles is* ***inconsistent****.*

(i) *T.*
(ii) *A1–A4.*
(iii) *A2–A6.*
(iv) *A3–A7.*

The natural conclusion to draw from these inconsistency results seems to be that we should abandon the predicate approach altogether. This has also been the conclusion drawn by many researchers within the knowledge representation community, and this is without doubt the reason that the operator approach is still so prevalent even though the predicate approach has a number of significant advantages (Section 2.4).[2] For our purpose—the construction of strongly introspective agents—the expressive power of the predicate approach is required, so rather than abandoning this approach we need to find a way to circumvent the inconsistency results that threatens it.

4.3. Regaining Consistency

In this section we will mention some of the possible ways to circumvent the inconsistency results presented in the previous section. An obvious possibility would be to choose an even smaller subset of the reflection principles than the ones shown to be inconsistent above. From the examples of Section 3.7.1 we know that even if we only take the reflection principles A3 and A4, our agents will still be able to infer at least *some* useful consequences of their knowledge. Another obvious possibility would be to modify the "form" of the reflection principles in order to make them weak enough to be consistent [Aczel and Feferman, 1980; Feferman, 1984; 1991; Friedman and Sheard, 1987; Grant *et al.*, 2000; McGee, 1985; Perlis, 1985; Turner, 1990]. A third possibility would be to replace the underlying logical framework (classical first-order predicate logic) by a weaker, non-standard logic such as for example a *many-valued logic* [Kerber, 1998], a *paraconsistent logic* [Priest, 1991; 1989] or a *typed logic*.

The strategy we will be choosing in this thesis to avoid the inconsistency problems is different from all of the above-mentioned. Our strategy is based on noting that all of the proofs of the inconsistency results above are based on reasoning about formal knower sentences. It therefore seems appropriate to somehow try to prevent such sentences from being reasoned about. There do not seem to be any particularly good reasons that an agent should be allowed to reason about pathologically self-referential sentences. Thus, if we can somehow prevent this from happening without at the same time sacrificing too much else, this would appear to be a good solution. Thus what we suggest is to try to exclude the pathological sentences from the reflection principles, that is, to only instantiate the reflection principles with sentences that are "safe" to reason about. This is the approach taken by des Rivières and Levesque [1988] and Morreau and Kraus [1998]. We will review their results

[2] It should be noted at this point that the operator approach has been shown to suffer from the same inconsistency problems when it is suitably extended with resources for encoding and substituting [Asher and Kamp, 1986; Perlis, 1988; Grim, 1993].

in the following. Rivières and Levesque show that the reflection principles can consistently be instantiated with what they call the *regular sentences* and Morreau and Kraus extend this result to the more inclusive class of what they call the *RPQ sentences*.

4.3.1. The Rivières-Levesque Theorem. The regular sentences are defined in the following way.

Definition 4.9 (After des Rivières & Levesque [1988]). *Let L be a first-order agent language. The set of* ***regular formulas*** *of L is the least set satisfying*

(i) *Any atomic formula of $L - \{K\}$ is a regular formula.*[3]
(ii) *The regular formulas are closed under the connectives $\wedge$, $\neg$ and $\forall$.*[4]
(iii) *If φ is a regular formula then $K\varphi$ is a regular formula.*[5]

A ***regular sentence*** *is a closed regular formula.*

The following result is proved by des Rivières and Levesque [1988].

Theorem 4.10 (Rivières-Levesque theorem). *Let L be a first-order agent language. Suppose S is the theory Q extended with one of the following sets of reflection principles*

(i) *The axiom schemes A1–A4 instantiated over the regular sentences of L.*
(ii) *The axiom schemes A2–A6 instantiated over the regular sentences of L.*

Then S is consistent.

This theorem should be contrasted with the theorems of Montague and Thomason (Theorem 4.5 and Theorem 4.6). The theorem above shows that consistency can be regained from these inconsistency results if we just refrain from instantiating the reflection principles with non-regular sentences. The usefulness of the result of course highly depends on how expressive regular sentences really are. It depends on whether the sentences we would like our agents to be able to express can be expressed as regular ones. We will look into this in the following.

What characterises the regular formulas? Condition (ii) of Definition 4.9 gives us that the regular formulas are closed under all the connectives of

[3] Recall that $L - \{K\}$ denotes the language L with the predicate symbol K removed.

[4] We take this to mean that if φ and ψ are regular formulas and x is a variable, then $\varphi \wedge \psi$, $\neg\varphi$ and $\forall x \varphi$ are regular formulas.

[5] For simplicity, we have again suppressed mentioning of the subscript of K (the first argument to K). Letting the subscript be explicit, the condition would read: If φ is a regular formula and τ is any term in L then $K_\tau \varphi$ is a regular formula.

predicate logic. Together with condition (i) this implies that any formula not containing the K predicate is regular. From condition (iii) we furthermore see that a formula φ can only be regular if it satisfies the following condition:

> For every subformula of φ on the form $K\tau$ we have $\tau = \ulcorner\psi\urcorner$ for some regular formula ψ.

Notice that this implies that no formula containing Kx as a subformula—where x is any variable—can be regular.

Let us try to see which of the formulas we have considered so far are regular. Below is a representative subset of these formulas.

(i) $K\,On\,(black, floor)$
(ii) $K\neg K\,On\,(striped, white)$
(iii) $K\,(\neg K\,On\,(striped, floor) \wedge \neg K \neg On\,(striped, floor))$
(iv) $K\neg\exists x\,(About\,(x, striped) \wedge Kx)$
(v) $K_{ole}\,\forall x\,(About\,(x, cool\ jazz) \wedge K_{sue}\,x \rightarrow K_{bill}\,x)$
(vi) $K_{john}\,\exists x\,(K_{bill}\,x \wedge \neg K_{john}\,x)$
(vii) $\forall x \forall y\,(IterateK\,(\varphi, n, x) \wedge y < n \rightarrow K_y\,(x))$
(viii) $\forall x_1 \forall x_2\,(Neg\,(x_1, x_2) \rightarrow \neg\,(Kx_1 \wedge Kx_2))$

It is easily seen that only (i)–(iii) are regular, since each of the other formulas contain a subformula $K_\tau x$ for some variable x (and term τ). This implies that the Rivières-Levesque theorem is not of much use for our purpose. One of the main reasons we chose the predicate approach was to be able to formulate sentences such as (iv)–(viii) that involve quantification over knowledge. However, the Rivières-Levesque theorem does not guarantee that we can consistently instantiate the reflection principles with any of these sentences. It only guarantees that we can consistently instantiate with (i)–(iii), but the predicate approach is not even needed to express such sentences (a propositional modal logic would suffice).

What we are looking for is a stronger consistency result which will show that the reflection principles can also consistently be instantiated with sentences such as (iv)–(viii). Such a consistency result will show that we can also safely allow agents to reason about these sentences. Our goal in this thesis is to provide such a strengthened consistency result. Another strengthened consistency result appears in Morreau and Kraus [1998], which we review in the following.

4.3.2. The Morreau-Kraus Theorem. Morreau and Kraus [1998] extend the Rivières-Levesque theorem by showing that consistency can still be retained if we instantiate the reflection principles over the larger set of *RPQ sentences*. Since the RPQ sentences are defined in a somewhat non-standard way, we will not try to give our own version of the definition, but simply quote the original one as it appears in Morreau and Kraus [1998]. First we must note that corresponding to our first-order agent languages, they define a language $\mathcal{L}_\alpha$ including the language of arithmetic and, for each $i > 1$, an $(i+1)$-place predicate symbol α^i. These predicate symbols correspond to our predicate symbol K. That is, $\alpha_a^1(\ulcorner\varphi\urcorner)$ is read as "φ is known (believed) by agent a". If $\varphi(x_1, \ldots, x_n)$ is a formula with $x_1, \ldots, x_n$ free, then $\alpha_a^{n+1}(\ulcorner\varphi\urcorner, x_1, \ldots, x_n)$ is intended to express that $\varphi(x_1, \ldots, x_n)$ is known. The idea is that this allows us to write for instance

$$\exists x \alpha^2_{john}(\ulcorner \mathit{Phone\text{-}number}(\mathit{bill}, x)\urcorner, x), \tag{4.11}$$

which is intended to express that John knows the phone number of Bill (or, more precisely, that there is a number x such that John knows x to be the phone number of Bill) [Perlis, 1985]. The reason for the extra x in the argument of the knowledge predicate α is that the variable x is not free in the term $\ulcorner \mathit{Phone\text{-}number}(\mathit{bill}, x)\urcorner$—this term is simply a numeral. Thus writing

$$\exists x \alpha^1_{john}(\ulcorner \mathit{Phone\text{-}number}(\mathit{bill}, x)\urcorner)$$

would not give the intended effect, since here the scope of the existential quantifier is the *closed* formula $\alpha^1_{john}(\ulcorner \mathit{Phone\text{-}number}(\mathit{bill}, x)\urcorner)$. Besides the language $\mathcal{L}_\alpha$, Morreau and Kraus considers the language $\mathcal{L}_0$, which is $\mathcal{L}_\alpha$ with the predicate symbols α^i removed.

Morreau and Kraus [1998] write: "*To begin we must once again slightly extend $\mathcal{L}_\alpha$. We add to $\mathcal{L}_\alpha$ the new monadic predicate symbols T and $\mathcal{P}$. Also, in addition to the variables $x, x_1, x_2, \ldots, y, \ldots,$ we add a countable stock of new variables $X, X_1, X_2, \ldots, Y, \ldots$.*

Intuitively speaking, the symbol $\mathcal{P}$ will pick out (Gödel numbers of) sentences of $\mathcal{L}_\alpha$; T will pick out true sentences. The following definition concerns the promised set, to which we shall generalize Theorem 3 [the Rivières-Levesque theorem].

***Definition** (Regular formulas with propositional quantification (RPQ)). The RPQ formulas are the smallest set such that:*

1. *Any atomic formula of $\mathcal{L}_0$ is an RPQ formula, as is $T(X)$, for any (new) variable X.*
2. *If φ and ψ are RPQ formulas and x is an (old) variable, then $\varphi \wedge \psi$, $\varphi \wedge \psi$, $\varphi \to \psi$, $\neg\varphi$, $\forall x \varphi$ and $\exists x \varphi$ are RPQ formulas.*
3. *If $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_l)$ is an RPQ formula, and a is a constant, then*
$$\alpha_a^{k+l+1}(\ulcorner\varphi\urcorner, x_1, \ldots, x_k, X_1, \ldots, X_l)$$
is an RPQ formula.
4. *If φ is an RPQ formula then $\forall X(\mathcal{P}(X) \to \varphi)$ and $\exists X(\mathcal{P}(X) \wedge \varphi)$ are also RPQ formulas, where X is a (new) variable.*"

It is easy to see that any regular formula is also an RPQ formula (when replacing K by α^1). Morreau and Kraus give the following example of an RPQ sentence

$$\forall X \left(\mathcal{P}(X) \rightarrow \left(\mathit{utter}(\mathit{system}, X) \rightarrow \alpha^2_{\mathit{system}}(\ulcorner T(X) \urcorner, X)\right)\right) \tag{4.12}$$

which expresses, in their words, that "the system says only what it takes [believes, knows] to be true". To express a sentence such as (v) above as an RPQ sentence, we would have to write

$$\alpha^1_{\mathit{ole}}\left(\ulcorner \forall X \left(\mathcal{P}(X) \rightarrow \left(\mathit{About}(X, \mathit{cool\ jazz}) \wedge \alpha^2_{\mathit{sue}}(\ulcorner T(X) \urcorner, X) \rightarrow \right.\right.\right.$$
$$\left.\left.\left.\alpha^2_{\mathit{bill}}(\ulcorner T(X) \urcorner, X)\right)\right)\right),$$

which is certainly more cumbersome. Conversely, in the context of our first-order agent languages, we would have expressed (4.12) as

$$\forall x \left(\mathit{Utter}(\mathit{system}, x) \rightarrow K_{\mathit{system}} x\right).$$

Morrau and Kraus [1998] prove the following consistency result.

Theorem 4.11 (Morreau-Kraus theorem). *Suppose S is the theory Q extended with one of the following sets of reflection principles*

(i) *The axiom schemes A1–A4 instantiated over the RPQ sentences of $\mathcal{L}_\alpha$.*
(ii) *The axiom schemes A2–A6 instantiated over the RPQ sentences of $\mathcal{L}_\alpha$.*

Then S is consistent.

Consider the two theories proven to be consistent by this theorem. There are no axioms in these theories that prevent us from constructing models in which $\mathcal{P}$ is assigned the extension $\emptyset$. In such models, a sentence as for example (4.12) would of course be trivially true and not receive its intended interpretation. The fact that we can construct models of the two theories in which $\mathcal{P}$ is assigned the empty extension, makes the Morreau-Kraus theorem an almost entirely trivial consequence of the Rivières-Levesque theorem. This is without doubt not what the two authors intended. The predicate symbol $\mathcal{P}$ is of course not intended to have the empty extension, and neither does it receive this extension in the model constructed by Morreau and Kraus.

Our work is related to the work of Morreau and Kraus in that we are also going to provide an extension of the Rivières-Levesque consistency result. We stay within our basic first-order agent languages, however, and do not extend the language with special variables, extra predicate symbols such as $\mathcal{P}$ and T or infinitely many predicate symbols K^i of different arities. These differences makes it problematic to give a precise mathematical comparison of our results with theirs. Adding to this that they have formulated their consistency result so that it is a direct consequence of the Rivières-Levesque theorem (by constructing a model I with $\mathcal{P}^I = \emptyset$), we will not be considering their result further in this thesis.

4.4. Strengthening the Consistency Results

As argued in Section 4.3.1, the Rivières-Levesque theorem is not sufficiently strong for our purposes. Most of the sentences we have considered so far are not regular. The Rivières-Levesque theorem guarantees that it is safe to instantiate the reflection principles with regular sentences, but does not guarantee that we can still retain consistency if we start instantiating with non-regular ones. If we are restricted to only instantiate the reflection principles with regular sentences, then our agents will only be able to reason about such sentences. This has the unfortunate consequence of preventing the agents from carrying out any kind of strongly introspective reasoning. Actually, if we are only interested in regular sentences, we could just as well choose the operator approach, since any regular sentence corresponds to a sentence within this modal formalism.

To allow the construction of strongly introspective agents on a safe foundation, that is, without being in risk of allowing paradoxical reasoning, we must obtain a much stronger consistency result than the Rivières-Levesque theorem. That is, we must find a considerably larger set M of sentences that the reflection principles can consistently be instantiated with. How can we find such a set? The Rivières-Levesque theorem is proved by a careful translation into predicate logic of a corresponding first-order modal logic. It is therefore not surprising that the result only concerns sentences that are expressible within this modal formalism. To strengthen their result, it seems that we are required to develop an alternative method.[6]

To find a larger set M over which we can safely instantiate our reflection principles, we have to take a closer look at the inconsistency results (Theorems 4.4, 4.5, 4.6 and 4.7). As noted previously, these are all based on instantiating reflection principles with formal knower sentences. So we need to exclude at least such sentences from the set M. At the same time, we wish to exclude as few sentences from M as possible, to get the strongest possible consistency result. To find such a set we are required to be able to *characterise* the "knower-like" sentences that we need to exclude from it. We could characterise these sentences *informally* by their self-referentiality. In the next chapter we will try to make this informal characterisation precise by providing

[6] That is, unless we want to try to make translations into predicate logic of higher-order modal formalisms, but we have not looked in this possibility. The Morreau-Kraus theorem is, however, obtained by such a translation from a second-order modal logic.

a *mathematical* characterisation of self-referentiality. This will be done in the following way.

First we associate with any first-order agent language a *dependency graph*. The nodes of this graph are the sentences in the language, and there is an edge from a sentence α to a sentence β if and only if α semantically *refers* to β. Given such a graph, we can then say that a sentence φ is *self-referential* if it is contained in a cycle in the dependency graph. In this way we are able to characterise the self-referential sentences and thereby the "knower-like" sentences that we have to exclude from M. This will then allow us to construct various sets M which are considerably larger than the set of regular sentences, but at the same time do not include any pathological "knower-like" sentences. We will prove that the reflection principles can consistently be instantiated over these sets of sentences.

4.5. Chapter Notes

Example 4.1 is the author's. A related modality of *only-knowing-about* is introduced in Levesque and Lakemeyer [2000]. Our proof of the diagonalisation lemma is roughly the proof given by Mendelson [1997]. We have included it because the diagonalisation lemma and how it is brought about is very important to the work of this thesis. It is the diagonalisation lemma that leads to all of the inconsistency results. Another reason for including the proof of the lemma has been to show that it uses equality in an essential way. This is of significance to our methods for regaining consistency. The discussion of the diagonalisation lemma is adapted from Bolander [2002c].

As mentioned above, our proofs of the theorems of Montague and Thomason are considerable simpler than the original proofs (this is made possible by having a slightly stronger version of the reflection principle A3). Among the advantages of these simpler proofs is that all of the theorems of Tarski, Montague and Thomason become based on the *same* self-referential sentence which we call a *formal knower sentence.* This makes it more clear what kind of sentences we have to exclude from our reflection principles to regain consistency.

The last inconsistency result, Theorem 4.7, is due to the author. It shows that even if we make absolutely no claims to the connection between what is believed and what is true, we still risk inconsistency.

Our definition of the *regular formulas* is adapted from the definition given by Morreau and Kraus [1998]. Des Rivières and Levesque [1988] give an alternative, but equivalent, definition of the regular formulas. Our definition is slightly simplified compared to the original definition. This also implies that our set of regular formulas is slightly more restrictive than the set of regular formulas considered by the aforementioned authors. We have chosen the alternative definition since it turns out to be more appropriate for our purposes. Everything we have said about the regular formulas in this chapter also holds for the regular formulas in the original formulation.

In this chapter we mentioned various approaches towards regaining consistency. We presented our own approach as based on the idea that to regain consistency we only need to prevent the pathological self-referential sentences to be reasoned about. This idea is rooted in the understanding that nothing useful can result from reasoning about pathological sentences such as formal knower sentences, so if we can just prevent such reasoning from taking place, we can keep everything else. The idea seems to be somewhat original, since usually more drastic methods are used to try to regain consistency: weakening the underlying logic or changing the forms of the reflection principles.

CHAPTER 5

The Graph Approach to Avoiding Inconsistency

In this chapter we will introduce our graph-based method, which is going to allow us to generalise the Rivières-Levesque theorem. The method is based on two new concepts: *sentence nets* and *dependency graphs*. These new concepts are defined in the beginning of the chapter. Subsequently we will show how proving the consistency of restricted reflection principles can be reduced to proving properties of corresponding dependency graphs. In the next chapter we will use this reduction to prove our two main results strengthening the Rivières-Levesque theorem.

5.1. Sentence Nets

We will not be associating dependency graphs directly with our first-order agent languages. Rather we will go through a simple formal language especially designed to express self-referential and related types of sentences. In this language, sets of mutually referring sentences can be expressed in a simple way. We will present this formal language in the following.

The idea behind the formal language we are about to present can be sketched as follows. Some sentences in natural language are characterised by only making claims about the truth or falsity of other sentences in that same language (as a special case, only making claims about the truth or falsity of themselves). Among the simplest examples is the sentence T saying of itself that it is true. We can express T in the following way

$$T : \text{sentence } T \text{ is true.} \tag{5.1}$$

Sentence T is known as the *truth-teller sentence*. We also have the *liar sentence* introduced in Section 1.4.1:

$$L : \text{sentence } L \text{ is not true.} \tag{5.2}$$

Both of T and L demonstrate cases of *direct self-reference*: sentences that directly make claims about themselves. We can also construct sentences that are *indirectly* self-referential as for instance in the following pair

$$\begin{aligned} s_1 &: \text{sentence } s_2 \text{ is true.} \\ s_2 &: \text{sentence } s_1 \text{ is false.} \end{aligned} \tag{5.3}$$

We are going to define a formal language for expressing such sentences in a simple way. This will allow us to give simple representations of various paradoxes such as the liar paradox and the knower paradox within the language. Through these representations we are then be able to study the paradoxes and their properties. Furthermore, we are able to investigate the conditions under which a set of sentences is *paradoxical*. This is of central importance to our search for consistent instantiation classes of the reflection principles.

Definition 5.1. *Let N be a countable set. We define a formal language $\mathscr{L}_N$ by*

(i) *For any $s \in N$, s is an **expression**.*
(ii) true *and* false *are **expressions**.*
(iii) *If E is an expression then $\neg E$ is an expression.*
(iv) *If $\{E_i \mid i \in M\}$ is a (possibly infinite) set of expressions, then both $\bigwedge \{E_i \mid i \in M\}$ and $\bigvee \{E_i \mid i \in M\}$ are expressions.*[1]
(v) *If s is an element of N and E is an expression, then $s : E$ is a **clause**. The element s is called the **head** of $s : E$ and E is called the **body**.*

*The elements of N are called **sentences** in $\mathscr{L}_N$.*

Remark. *In the following we will, unless otherwise stated, assume N to be denoting a* fixed *countable set.*

The language $\mathscr{L}_{\{s_i \mid i \in \mathbb{N}\}}$ contains clauses such as

$$s_1 : s_2 \vee s_3 \tag{5.4}$$

and

$$s_1 : \bigvee \{\neg s_i \mid i > 1\}. \tag{5.5}$$

The intended meaning of the clause (5.4) is that the sentence s_1 claims that either s_2 or s_3 is true. It is simply a short-hand for

$$s_1 : s_2 \text{ is true or } s_3 \text{ is true.}$$

Clause (5.5) is a short-hand for

$$s_1 : \text{one of the } s_i \text{ with } i > 1 \text{ is not true.}$$

[1]We sometimes write $\bigwedge_{i \in M} E_i$ and $\bigvee_{i \in M} E_i$ instead of $\bigwedge \{E_i \mid i \in M\}$ and $\bigvee \{E_i \mid i \in M\}$, respectively. As usual, we most often use infix notation in the finite case, that is, we write $E_1 \wedge E_2 \wedge \cdots \wedge E_n$ instead of $\bigwedge \{E_i \mid 1 \le i \le n\}$ and similarly for $\bigvee$.

As further examples of clauses in $\mathscr{L}_{\{s_i|i\in\mathbb{N}\}}$ we have

$$\begin{array}{ll} s_1 : \neg s_1 & s_1 \text{ is the liar sentence} \\ s_2 : s_2 & s_2 \text{ is the truth-teller sentence} \\ \left.\begin{array}{l} s_3 : \neg s_4 \\ s_4 : s_3 \end{array}\right\} & s_3 \text{ and } s_4 \text{ are mutually referring sentences} \end{array} \tag{5.6}$$

When we work with sets of clauses we want to avoid the case where two different clauses have the same head. Sets of clauses satisfying this requirement are called *sentence nets*.

Definition 5.2. *Let N be a countable set. A* ***sentence net*** *U over N is a set of clauses in $\mathscr{L}_N$ in which no two clauses have the same head. The elements of N are called* ***sentences*** *in U.*

Note that this definition ensures that when considering sentence nets we always have a canonical isomorphism between clauses and sentences: the isomorphism that maps every clause to its head. We will therefore often be identifying clauses with their corresponding heads.

Example 5.3. The four clauses (5.6) together form a sentence net over $\{s_1, s_2, s_3, s_4\}$. The set $\{s_1 : s_2, s_1 : s_3\}$, however, is not a sentence net since there are two clauses with the same head.

We will now define the semantics for sentence nets.

Definition 5.4. *Let U be a sentence net over a countable set N. An* ***interpretation*** *of U is a map I from N to the set of truth-values $\{t, f\}$. Interpretations extend to give truth-values to the expressions* true *and* false *by letting $I(\text{true}) = t$ and $I(\text{false}) = f$. Interpretations extend further to give truth-values to all compound expressions in $\mathscr{L}_N$ in the obvious, classical way. An interpretation $I : N \to \{t, f\}$ is called a* ***model*** *of U if for every clause $s : E$ in U we have*

$$I(s) = I(E). \tag{5.7}$$

Example 5.5 (The liar and the truth-teller)**.** Let U_t be the sentence net over $\{s\}$ consisting only of the clause $s : s$. The sentence s is the *truth-teller sentence*. It is a sentence s to the effect that s itself is true. It thus corresponds to the sentence "this sentence is true". U_t has the two interpretations I_1 and I_2 given by $I_1(s) = t$ and $I_2(s) = f$. These are both models of U_t, since equation (5.7) in this case simply reduces to $I(s) = I(s)$. The existence of these two models corresponds to the fact that we can consistently assign both the truth-value *true* and the truth-value *false* to the sentence "this sentence is true".

Let U_l be the sentence net consisting only of the clause $s : \neg s$. The sentence s is the *liar sentence*. The clause $s : \neg s$ is short-hand for

$$s : s \text{ is not true.}$$

The sentence defined by the clause is a sentence s to the effect that s itself is not true. It thus corresponds to the sentence "this sentence is not true". U_l has the same two interpretations as U_t, but it has no models since the equation $I(s) = I(\neg s)$ does not hold for any interpretation I ($\neg s$ receives the opposite truth-value of s under I). This corresponds to the fact that the sentence "this sentence is not true" can neither consistently be assigned the truth-value *true* nor the truth-value *false*. It is a paradoxical sentence (the *liar paradox*).

Definition 5.6. *A sentence net is called* ***paradoxical*** *if it does not have a model. Otherwise it is called* ***non-paradoxical****.*

Example 5.7 (The liar and the truth-teller)**.** In Example 5.5 we considered the sentence nets $U_t = \{s : s\}$ and $U_l = \{s : \neg s\}$. The sentence net U_t has a model, so it is non-paradoxical. The sentence net U_l does not have a model, so it is paradoxical. The paradoxicality of U_l corresponds exactly to the **liar paradox** introduced in Section 1.4.1. It is the paradoxicality of the sentence s to the effect that s itself is not true.

In Section 5.3 we will be showing how to associate a canonical sentence net with any first-order agent language. In Section 5.4 this will be used to show how the question of consistency of first-order agent theories is related to the question of paradoxicality of corresponding sentence nets. Before we turn to that, we will show how to associate *dependency graphs* with sentence nets.

5.2. Dependency Graphs

Our current situation can be sketched as follows: We are looking for a way to characterise paradoxicality and self-reference that will allow us to strengthen the Rivières-Levesque theorem. To this end we now introduce *dependency graphs*. Dependency graphs are graphs associated with sentence nets. The dependency graph of a sentence net is intended to represent the *patterns of semantic dependency* among the sentences in the net. If s_1 and s_2 are sentences in the net, then there will be an edge from s_1 to s_2 in the dependency graph if and only if the sentence s_1 contains a semantic reference to s_2. This is the case for instance with the sentence s_1 in (5.3) on page 86. Dependency graphs are defined in the following way.

Definition 5.8. *Let U be a sentence net over a countable set N. The* ***dependency graph*** *of U is the directed graph G given by*

- *Every sentence in U is a node in G.*

FIGURE 5.1. The simple loop.

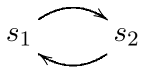

FIGURE 5.2. Simple indirect self-reference.

- *If $s_1 : E$ is a clause in U and s_2 is a sentence occurring in E, then there is an edge from s_1 to s_2 in G.*

EXAMPLE 5.9 (The liar)**.** Consider again the sentence net

$$U_l = \{s : \neg s\}.$$

The sentence s is the *liar sentence.* The dependency graph of U_l is presented in Figure 5.1. The graph is called the **simple loop**. The sentence net $U_t = \{s : s\}$ also has the simple loop as its dependency graph.

EXAMPLE 5.10. Consider the sentence net over $\{s_1, s_2\}$ consisting of the following two clauses

$$s_1 : \neg s_2$$
$$s_2 : s_1.$$

The dependency graph of this sentence net is presented in Figure 5.2. It is a cycle spanned by the two nodes s_1 and s_2.

Graphs are usually given as pairs $G = (V, B)$, where V is the set of nodes and $B \subseteq V^2$ is the *edge relation* of G. The dependency graph of a sentence net U over N can thus be expressed as

$$G = \big(N, \{(s_1, s_2) \mid s_1 : E \text{ is in } U \text{ and } s_2 \text{ occurs in } E\}\big).$$

The edge relation of the dependency graph of U is called the **dependency relation** of U. We will generally be identifying dependency graphs with their corresponding dependency relations. A graph is **countable** when its set of nodes is either a finite or a countably infinite set.

REMARK. *By **graph** we will everywhere mean a directed and countable graph.*

A **path** in a graph G is a sequence $(r_1, r_2, \ldots, r_n)$ of nodes such that (r_i, r_{i+1}) is an edge in G for all $0 < i < n$. The path is called **simple** if all the nodes on the path are distinct. If $r_1 = r_n$ then the path is called a **cycle**.

Any graph containing a cycle is called a **cyclic graph**. Otherwise it is called **acyclic**. If (r_1, r_2) is an edge in a graph and $r_1 = r_2$, then the edge is called a **loop** at r_1. An **infinite path** in G is an infinite sequence $(r_1, r_2, r_3, \dots)$ of nodes such that (r_i, r_{i+1}) is an edge in G for all $i > 0$. Note that if $(r_1, \dots, r_n)$ is a cycle, then

$$(r_1, \dots, r_{n-1}, r_1, \dots, r_{n-1}, r_1, \dots, r_{n-1}, \dots)$$

is an infinite path. Thus if a node is contained in a cycle, then it is also contained in an infinite path. If $(r_1, r_2, \dots)$ is a finite or infinite path, then any subsequence of $(r_1, r_2, \dots)$ is called a **subpath**. If (r_1, r_2) is an edge, then r_1 is called the **start node** and r_2 the **end node** of the edge. If s is a node in a graph $G = (V, B)$, then we use $\Gamma_G(s)$ to denote the set of successors of s in G. That is, we define $\Gamma_G(s)$ by

$$\Gamma_G(s) = \left\{ s' \mid (s, s') \in B \right\}.$$

If G is the dependency graph of a sentence net U and $s : E$ is a clause in U, then $\Gamma_G(s)$ is the set of sentences occurring in E. A graph G is called **well-founded** if it does not contain any infinite paths. Otherwise it is called **non-wellfounded**. Note that any well-founded graph is acyclic, since if a graph does not contain any infinite paths then it can in particular not contain any cycles.

The idea of representing semantic dependency between sentences by graphs is by no means new. Probably the area where this idea has been pursued to the largest extend is in *logic programming*. In this field, dependency graphs where originally introduced by Apt *et al.* [1988], and since then they have been used extensively to prove the existence of various types of semantics for general logic programs (see for instance Apt and Bol [1994] or Baral and Gelfond [1994] for surveys). Our dependency graphs are very closely related to the ones used in logic programming, and we will study these relationships in detail in Section 5.5. Dependency graphs have also been studied in philosophical logic. In the literature on truth predicates and paradoxes, dependency graphs have been used explicitly in among others Beck [2002], Cook [2002], Gaifman [1992] and Yablo [1982]. Another use of graphs related to the present work is the one found in non-wellfounded set theory [Aczel, 1988; Barwise and Etchemendy, 1987; Barwise and Moss, 1996].[2]

[2] We should note that the term *dependency graph* also occurs in connection with graphs used to model data and control flow in hardware and software design. In this thesis, we only use the term *dependency graph* in the sense of graphs representing semantic dependencies between formal sentences.

We are now ready to define what it means for a sentence in a sentence net to be self-referential. The definition uses the connection to dependency graphs.

Definition 5.11. *Let U be a sentence net and let G be its dependency graph. Let s_1 and s_2 be sentences in U. We say that*

- *s_1 **refers directly** to s_2 in U if (s_1, s_2) is an edge in G.*
- *s_1 **refers indirectly** to s_2 in U if there is a path from s_1 to s_2 in G.*
- *s_1 is **directly self-referential** in U if G contains a loop at s_1.*
- *s_1 is **indirectly self-referential** in U if there is a cycle in G containing s_1.*
- *s_1 is **doubly dependent** on s_2 in U if there are at least two distinct paths from s_1 to s_2 in G.*

Example 5.12 (The liar)**.** Consider again the sentence net

$$U_l = \{s : \neg s\}.$$

The dependency graph of this sentence net is the simple loop presented in Figure 5.1. The loop represents the fact that the liar sentence is *directly self-referential.* Note that, in addition, s is *doubly dependent* on itself since both (s, s) and (s, s, s) are paths from s to s.

Example 5.13. Consider again the sentence net over $\{s_1, s_2\}$ consisting of the following two clauses

$$s_1 : \neg s_2$$
$$s_2 : s_1.$$

The dependency graph of this sentence net was presented in Figure 5.2. From this graph we see that the first sentence, s_1, *refers directly* to the second, s_2, represented by the edge from s_1 to s_2. The second sentence *refers directly* to the first, represented by the edge from s_2 to s_1. We note that the graph is cyclic: there is a path leading from s_1 through s_2 back to s_1. Thus both sentences are *indirectly self-referential.*

Definition 5.14. *A graph G is said to be **paradoxical** if there exists a paradoxical sentence net having G as its dependency graph. Otherwise G is called **non-paradoxical**.*

Using the definition of a paradoxical sentence net (Definition 5.6), we see that a graph G is non-paradoxical if and only if every sentence net with dependency graph G has a model. Note that we are using the notions *paradoxical* and *non-paradoxical* both for sentence nets and for graphs. Suppose U is a sentence net and G is its dependency graph. If U is paradoxical then G is paradoxical as well, but the opposite is not necessarily true. That is, G can be paradoxical without U being paradoxical. If G is paradoxical it just means that there is *some* paradoxical sentence net with dependency graph G—this could be another sentence net than U.

Example 5.15. The simple loop considered in Example 5.9 is the dependency graph of both of the sentence nets U_l and U_t. In Example 5.7, we saw that U_l is paradoxical while U_t is non-paradoxical. The paradoxicality of U_l implies that the simple loop is paradoxical.

In the following section we will show how we can associate sentence nets (and thus, indirectly, dependency graphs) with first-order agent languages. Through this association we become able to use the notions defined in Definition 5.11 directly on sentences of agent languages. In particular, we get a notion of when a sentence in an agent language is self-referential.

5.3. From Agent Languages to Dependency Graphs

Remark. *To simplify notation, we will take L to denote a* fixed *first-order agent language in the remainder of this thesis.*

5.3.1. The Dependency Graph G_L.

The following definition shows how we associate a "canonical" sentence net U_L with the first-order language L. In this way a "canonical" dependency graph will by associated with L as well: the dependency graph of the sentence net U_L.

Definition 5.16. *We define U_L to be the sentence net over $\{s_\varphi \mid \varphi \in L\}$ consisting of the following clauses*

(i)	$s_{\varphi\wedge\psi} : s_\varphi \wedge s_\psi,$	*for all sentences φ, ψ in L.*
(ii)	$s_{\neg\varphi} : \neg s_\varphi,$	*for all sentences φ in L.*
(iii)	$s_{\forall x\varphi(x)} : \bigwedge_{\tau\in\text{Terms}(L)} s_{\varphi(\tau)},$	*for all variables x and all $\forall x\varphi(x)$ in L.*
(iv)	$s_{K\varphi} : s_\varphi,$	*for all sentences φ in L.*
(v)	$s_\varphi : \text{true},$	*when φ is atomic in L and $Q \vdash \varphi$.*
(vi)	$s_\varphi : \text{false},$	*when φ is atomic in L and $Q \vdash \neg\varphi$.*

The dependency graph of U_L is denoted G_L.

The nodes of G_L are sentences of the form s_φ where φ is a sentence in L. To simplify notation, we will be identifying every node s_φ in G_L with the corresponding first-order sentence φ. Thus, by the identification, the nodes of G_L are the sentences of L. From items (i)–(iv) in Definition 5.16 we see that G_L contains the following set of edges

(I)	$(\varphi\wedge\psi, \varphi)$ and $(\varphi\wedge\psi, \psi)$,	for all sentences φ, ψ in L.
(II)	$(\neg\varphi, \varphi)$,	for all sentences φ in L.
(III)	$(\forall x\varphi(x), \varphi(\tau))$,	for all $\forall x\varphi(x)$ in L and all terms τ in L.
(IV)	$(K\varphi, \varphi)$,	for all sentences φ in L.

$$\begin{tikzcd}
 & K\,(\neg K\,On\,(\textit{striped}, \textit{dotted}) \wedge \neg K \neg On\,(\textit{striped}, \textit{dotted})) \arrow[d, "K"] & \\
 & \neg K\,On\,(\textit{striped}, \textit{dotted}) \wedge \neg K \neg On\,(\textit{striped}, \textit{dotted}) \arrow[dl, "\wedge"'] \arrow[dr, "\wedge"] & \\
\neg K\,On\,(\textit{striped}, \textit{dotted}) \arrow[d, "\neg"] & & \neg K \neg On\,(\textit{striped}, \textit{dotted}) \arrow[d, "\neg"] \\
K\,On\,(\textit{striped}, \textit{dotted}) \arrow[d, "K"] & & K \neg On\,(\textit{striped}, \textit{dotted}) \arrow[d, "K"] \\
On\,(\textit{striped}, \textit{dotted}) & & \neg On\,(\textit{striped}, \textit{dotted}) \arrow[ll, "\neg"]
\end{tikzcd}$$

FIGURE 5.3. A subgraph of G_L.

We will be referring to these edges by the names given in the following definition.

DEFINITION 5.17. *Any pair of the form $(\varphi \wedge \psi, \varphi)$ or $(\varphi \wedge \psi, \psi)$, where φ and ψ are sentences of L, is called a **$\wedge$-edge**. A pair of the form $(\neg\varphi, \varphi)$ is called a **$\neg$-edges** and a pair of the form $(K\varphi, \varphi)$ a **K-edge**. Finally, a pair of the form $(\forall x \varphi(x), \varphi(\tau))$, where $\forall x \varphi(x)$ is a sentence in L and τ is a term in L, is called a **$\forall$-edges**.*

By (I)–(IV) above we see that any edge in G_L is of one of the four types given in this definition (when identifying s_φ with φ for all φ in L). The underlying intuition here is that we have $\wedge$-edges to represent the fact that any sentence $\varphi \wedge \psi$ refers directly to both φ and ψ; $\neg$-edges to represent the fact that any sentence $\neg\varphi$ refers directly to φ; $\forall$-edges to represent the fact that any sentence $\forall x \varphi(x)$ refers directly to each substitution instance $\varphi(\tau)$; and finally K-edges to represent the fact that any sentence $K\varphi$ refers directly to φ.

EXAMPLE 5.18. Let us try to look at a fragment (subgraph) of G_L. Let φ be the sentence given by

$$\varphi = K\,(\neg K\,On\,(\textit{striped}, \textit{dotted}) \wedge \neg K \neg On\,(\textit{striped}, \textit{dotted}))\,.$$

If φ is a sentence in L, then G_L will contain the subgraph presented in Figure 5.3, where we have marked each edge by its type. From this subgraph we see that φ *refers indirectly* to $On\,(\textit{striped}, \textit{dotted})$ (in U_L). It is even *doubly dependent* on this subformula. The sentence φ is not *self-referential*, since the

$$
\begin{tikzcd}
 & \psi \arrow[d, "K"] & \\
 & \forall x \neg \left(About\,(x, striped) \wedge Kx \right) \arrow[d, "\forall"] & \\
 & \neg \left(About\,(\psi, striped) \wedge K\psi \right) \arrow[d, "\neg"] & \\
 & About\,(\psi, striped) \wedge K\psi \arrow[dl, "\wedge"'] \arrow[dr, "\wedge"] & \\
About\,(\psi, striped) & & K\psi \arrow[uuuul, bend right, "K"']
\end{tikzcd}
$$

FIGURE 5.4. Another subgraph of G_L.

$$
\begin{tikzcd}
 & \neg \left(About\,(\psi, striped) \wedge K\psi \right) \arrow[dl, "\neg"'] & \\
About\,(\psi, striped) \wedge K\psi \arrow[d, "\wedge"] & \circlearrowleft & \forall x \neg \left(About\,(x, striped) \wedge Kx \right) \arrow[ul, "\forall"'] \\
K\psi \arrow[rr, "K"] & & \psi \arrow[u, "K"']
\end{tikzcd}
$$

FIGURE 5.5. A cycle in G_L.

subgraph shown includes all paths starting at φ, and none of these paths are cycles. These conclusions about φ all fit well with our intuitions.

EXAMPLE 5.19. Consider the sentence

$$K \forall x \left(About\,(x, striped) \rightarrow \neg Kx \right).$$

This is an abbreviation for the sentence ψ given by

$$\psi = K \forall x \neg \left(About\,(x, striped) \wedge Kx \right).$$

If ψ is a sentence in L, then G_L will contain the subgraph presented in Figure 5.4. From this subgraph we see that ψ *refers indirectly* to $About\,(\psi, striped)$. We also see that ψ must be *indirectly self-referential*, since the subgraph contains the cycle presented in Figure 5.5. This is in agreement with the conclusion we reached in Example 4.1. There we showed how the self-referentiality of ψ could cause a knowledge base containing it to become inconsistent.

$$
\begin{tikzcd}
& \beta \arrow[d, "\forall"] & \\
& \neg\left(D(\alpha,\beta) \wedge K\beta\right) \arrow[d, "\neg"] & \\
& D(\alpha,\beta) \wedge K\beta \arrow[dl, "\wedge"'] \arrow[dr, "\wedge"] & \\
D(\alpha,\beta) & & K\beta \arrow[uuul, bend right, "K"']
\end{tikzcd}
$$

FIGURE 5.6. Yet another subgraph of G_L.

EXAMPLE 5.20. As a final example of a subgraph of G_L, consider a *formal knower sentence* given by

$$\forall x_2 \left(D(\alpha, x_2) \rightarrow \neg K(x_2)\right), \tag{5.8}$$

where $\alpha = \forall x_2 \left(D(x_1, x_2) \rightarrow \neg K\left(x_2\right)\right)$ (cf. the diagonalisation lemma, Lemma 4.2). The sentence (5.8) is an abbreviation for the sentence β given by

$$\beta = \forall x_2 \neg \left(D(\alpha, x_2) \wedge K(x_2)\right).$$

The sentence β is contained in the subgraph of G_L presented in Figure 5.6. The sentence β is contained in a cycle in this subgraph. Thus β is *indirectly self-referential*, as expected.

We have indicated that the edges of G_L should be interpreted as representing relations of *reference*. Alternatively, we can think of the edges as representing relations of *semantic dependency*. In the standard Tarskian (truth-functional) semantics, the truth-value of a conjunction $\varphi \wedge \psi$ is defined uniquely in terms of the truth-values of the conjuncts φ and ψ. This fact is represented in G_L by having an edge from $\varphi \wedge \psi$ to each of the conjuncts φ and ψ. Similarly, we have an edge from $\neg\varphi$ to φ to represent the fact that the truth-value of $\neg\varphi$ depends uniquely on the truth-value of φ. In the Tarskian semantics, the truth-value of a sentence $\forall x \varphi(x)$ is determined uniquely in terms of the truth-values of the infinitely many substitution instances $\varphi(\tau)$ where $\tau \in \mathrm{Terms}(L)$ (assuming that the domain of the interpretation is $\mathrm{Terms}(L)$). This fact is represented by having an edge from $\forall x \varphi(x)$ to each $\varphi(\tau)$ where $\tau \in \mathrm{Terms}(L)$. In general, we can think of any edge (φ, ψ) in the graph as expressing: "The semantic value of φ depends directly on the semantic value of ψ". This is also the case with the special K-edges, which expresses that the semantic value of $K\varphi$ depends directly on the semantic value of φ. If a sentence φ is contained in a cycle in the graph, then this implies that φ depends semantically on *itself*. This, once again, corresponds to *self-reference*.

If we did not have the special K-edges in the graph, then we could be certain that the graph would not contain any cycles. This follows immediately from the fact that for any $\wedge$-, $\neg$- or $\forall$-edges (φ, ψ), the syntactic complexity of ψ is lower than the syntactic complexity of φ. Therefore, along any path in such a graph, the syntactic complexity would be decreasing, and the graph would thus be *well-founded* (see the proof of Lemma 5.21 below). The $\wedge$-, $\neg$- and $\forall$-edges adhere to the *principle of compositionality* in semantics: the semantic value of a compound expression depends uniquely on the semantic values of the constituents of the expression. The K-edges do *not* adhere to this principle. Consider a K-edge $(K\varphi, \varphi)$. The sentence φ is not a constituent in $K\varphi$, since the latter sentence is simply an abbreviation for $K(\ulcorner\varphi\urcorner)$, where $\ulcorner\varphi\urcorner$ is a numeral. The start node $K\varphi$ of the edge is an *atomic* sentence, whereas the end node φ can be any sentence. This implies that along K-edges the syntactic complexity will be (monotonically) *increasing* rather than *decreasing*. Thus with the presence of K-edges in the graph, we loose the possibility of semantically stratifying the sentences according to syntactic complexity. It is precisely this phenomenon that makes it difficult to give a semantics for the agent languages that takes proper care of the sentences involving the K predicate. As we have seen from the inconsistency results (Chapter 4), this difficulty sometimes even takes the form of an impossibility.

5.3.2. Basic Properties of G_L. As mentioned in Section 4.4, our goal is to use dependency graphs to see which instances of the reflection principles we need to exclude in order to regain consistency from the inconsistency results. As we will show in Section 5.4, this problem can be reduced to determining which K-edges we have to remove from G_L in order for the graph to become non-paradoxical. It seems that we are required to exclude at least all instances over self-referential sentences, since these are the ones causing the inconsistency results (Section 4.2). Suppose φ is an *indirectly self-referential* sentence in U_L. Then by definition, φ is contained in a *cycle* in G_L. From this it follows that φ must be contained in an *infinite path* in G_L. In the following, we will prove a number of general results concerning the kind of sentences and edges that are contained in infinite paths in G_L. This will then also give us information on the kind of sentences and edges to be found in *cycles* in G_L. From this we will be able to deduce valuable information on the properties of self-referential sentences, which will later become useful in the search for consistent sets of instances of our reflection principles.

Lemma 5.21. *Suppose σ is an infinite path in G_L. Then σ contains infinitely many K-edges.*

Proof. Assume to obtain a contradiction that G_L contains an infinite path σ with only finitely many K-edges. Then there will be an infinite subpath

σ' of σ with no K-edges. Thus all edges on this subpath must be $\wedge$-, $\neg$- or $\forall$-edges. That is, all edges must be on one of the following forms

(i) $(\varphi \wedge \psi, \varphi)$ or $(\varphi \wedge \psi, \psi)$, for some $\varphi, \psi \in L$.
(ii) $(\neg\varphi, \varphi)$, for some $\varphi \in L$.
(iii) $(\forall x \varphi(x), \varphi(\tau))$, for some $\forall x \varphi(x) \in L$ and $\tau \in \mathrm{Terms}(L)$.

Now note that on any such edge, the end node has lower syntactic complexity than the start node (that is, the end node contains fewer connectives than the start node). Thus along the subpath σ', the syntactic complexity will be strictly decreasing. But this contradicts that σ' is an infinite path, and the proof is hereby complete. $\square$

Example 5.22. Consider the subgraph of G_L presented in Figure 5.5. In this subgraph, ψ is the sentence

$$K \forall x \neg \left(\mathit{About}\,(x, \mathit{striped}) \wedge Kx \right).$$

The subgraph is a cycle and thus becomes an infinite path when repeatedly traversing the cycle. The cycle contains a K-edge, so the corresponding infinite path will contain infinitely many K-edges, in correspondence with Lemma 5.21.

Lemma 5.21 implies that if we remove all K-edges from G_L, then the graph will no longer contain any infinite paths and thus no longer any cycles. This would therefore be an effective cure against self-reference, but then any sentence $K\varphi$ would become semantically detached from φ, which is not our intention. The semantical link between $K\varphi$ and φ is needed in our pursuit of models of the reflection principles.

For the next lemma we first need a couple of new definitions.

Definition 5.23. *To every formula φ in L we associate a natural number $d(\varphi)$, called its* ***K-depth****. The function d is defined recursively by*

(i) $d(\varphi) = 0$*, if φ is an atomic sentence* not *on the form $K\alpha$ for some α in L.*
(ii) $d(K\varphi) = 1 + d(\varphi)$.
(iii) $d(\varphi \wedge \psi) = \max\{d(\varphi), d(\psi)\}$.
(iv) $d(\neg\varphi) = d(\varphi)$.
(v) $d(\forall x \varphi) = d(\varphi)$.

We need to check that the function d defined in this way is well-defined. To see this, note that we have defined our Gödel numbering (Definition 3.4) in such a way that the following holds for all formulas φ and ψ of predicate logic

$$\ulcorner\varphi \wedge \psi\urcorner > \ulcorner\varphi\urcorner, \ulcorner\psi\urcorner.$$
$$\ulcorner\neg\varphi\urcorner > \ulcorner\varphi\urcorner.$$
$$\ulcorner\forall x\varphi\urcorner > \ulcorner\varphi\urcorner.$$
$$\ulcorner K(\ulcorner\varphi\urcorner)\urcorner > \ulcorner\varphi\urcorner.$$

Thus, for every formula φ, the K-depth of φ is defined exclusively in terms of the K-depth of formulas with smaller Gödel numbers. This ensures that d is well-defined: $d(\varphi)$ is definable by recursion on the Gödel number of φ.

Example 5.24. The K-depth of any formula not containing the predicate symbol K is zero. The K-depth of $K(0 = 0)$ is one and the K-depth of $K(K(0 = 0))$ is two. The K-depth of $A(K(0 = 0))$ is zero if A is a predicate symbol different from K.

Definition 5.25. *A sentence φ in L is called **grounded** if it does not contain Kx as a subformula for any variable x. Otherwise it is called **ungrounded**.*[3]

Example 5.26. The sentence $K\neg K\, On\,(striped, white)$ is grounded, but $\forall x\,(About\,(x, striped) \rightarrow \neg Kx)$ is not. The sentence

$$K\forall x\,(About\,(x, striped) \rightarrow \neg Kx)$$

is also grounded, since it is simply an abbreviation for

$$K(\ulcorner\forall x\,(About\,(x, striped) \rightarrow \neg Kx)\urcorner).$$

We should note that our use of the term "grounded" is borrowed from Kripke [1975] rather than from the use of this term in logic programming. The intuition behind our *grounded sentences* is more or less the same as Kripke's, although the concept defined here is not identical to his.

We noted in Section 4.3.1 that if φ is a *regular sentence* then it does not contain Kx as a subformula for any variable x. Thus all regular sentences are grounded. The opposite is not the case. Consider the following sentence

$$K\forall x Kx.$$

This sentence is an abbreviation for

$$K\left(\ulcorner\forall x K(x)\urcorner\right),$$

so it does not contain Kx as a subformula. It is therefore grounded, but it is not regular, because then $\forall x Kx$ would have to be regular as well. If n is

[3]We still suppress mentioning of the subscript of K. To be more precise, φ is grounded if it does not contain $K_\tau x$ as a subformula for any variable x and any term τ.

a numeral not denoting any sentence in L, then Kn is also grounded but not regular. We therefore have the following result.

Lemma 5.27. *The set of grounded sentences of L is a proper extension of the set of regular sentences of L.*

In Section 6.1.1 we will show that all reflection principles can consistently be instantiated over the grounded sentences. Since the grounded sentences properly include the regular ones, this result will be a (minor) strengthening of the Rivières-Levesque theorem. Before we can prove this strengthened theorem we need to introduce some more machinery.

Lemma 5.28. *Suppose σ is an infinite path in G_L. Then σ contains infinitely many $\forall$-edges $(\forall x \varphi(x), \varphi(\tau))$, where Kx occurs in $\varphi(x)$.*

Proof. Assume to obtain a contradiction that there exists an infinite path σ in G_L with only finitely many $\forall$-edges $(\forall x \varphi(x), \varphi(\tau))$ for which Kx occurs in $\varphi(x)$. Then there will be an infinite subpath σ' of σ with no such edges.

Claim. The K-depth is monotonically decreasing along σ'.

Proof of claim. Let (r_1, r_2) be any edge in σ'. We have to show that $d(r_2) \leq d(r_1)$. If (r_1, r_2) is a K-, $\wedge$- or $\neg$-edge, we immediately get $d(r_2) \leq d(r_1)$ from the clauses (ii)–(iv) in the definition of d. The only case left to check is if (r_1, r_2) is a $\forall$-edge. So assume $r_1 = \forall x \varphi(x)$ and $r_2 = \varphi(\tau)$ for some formula φ and term τ. By choice of σ', the sentence $\forall x \varphi(x)$ does not have any occurrence of Kx. Thus we have $d(\varphi(x)) = d(\varphi(\tau))$, and therefore

$$d(r_1) = d(\forall x \varphi(x)) = d(\varphi(x)) = d(\varphi(\tau)) = d(r_2).$$

This concludes the proof of the claim.

From this claim we can now easily obtain a contradiction with our assumption. We have constructed a path σ', along which the K-depth is monotonically decreasing. Therefore the K-depth must be constant from some point. But then from this point the path can not contain any K-edges, since the K-depth of the end node of such an edge is always one less than the K-depth of the start node. This immediately contradicts Lemma 5.21. $\square$

Example 5.29. Consider again the cycle in Figure 5.5. The cycle contains the edge

$$\big(\forall x \neg \left(\mathit{About}\left(x, \mathit{striped}\right) \wedge Kx \right), \neg \left(\mathit{About}\left(x, \mathit{striped}\right) \wedge Kx \right)\big).$$

The start node of this edge contains the subformula Kx. It is thus the kind of edge guaranteed to exist in any infinite path (and cycle) by the lemma above. The lemma proves that it is impossible to obtain self-reference without using formulas in which Kx appears as subformula for some variable x—since any cycle will contain at least one such formula.

A direct consequence of Lemma 5.28 is that any infinite path in G_L will contain infinitely many ungrounded sentences.

LEMMA 5.30. *Let σ be an infinite path in G_L. Then σ contains infinitely many K-edges $(K\varphi, \varphi)$, where φ is ungrounded.*

PROOF. It suffices to prove that any infinite path contains at least one K-edge $(K\varphi, \varphi)$ where φ is ungrounded. Let thus an arbitrary infinite path σ be given. By Lemma 5.21, σ contains infinitely many K-edges. Let $\sigma' = (r_1, r_2, \dots)$ be an infinite subpath of σ in which the first edge (r_1, r_2) is a K-edge. By Lemma 5.28, the path σ' contains infinitely many $\forall$-edges $(\forall x\varphi(x), \varphi(\tau))$ where $\varphi(x)$ contains Kx. Let (r_k, r_{k+1}) be one of these edges. Then r_k contains Kx. Let (r_m, r_{m+1}) be the K-edge closest to r_k in the initial subpath $(r_1, \dots, r_k)$ of σ'. Since the first edge in σ' is a K-edge, we are guaranteed that such an edge exists. By the choice of m, there are no K-edges between r_{m+1} and r_k in σ'. This implies that all edges between these two nodes are $\wedge$-, $\neg$- or $\forall$-edges. This, in turn, implies that r_k must be a substitution instance of a subformula of r_{m+1}. Since r_k contains Kx as a subformula, r_{m+1} must then contain Kx as well. Thus r_{m+1} is ungrounded. This means that (r_m, r_{m+1}) is a K-edge on the form $(K\varphi, \varphi)$, where φ is ungrounded. Since $(K\varphi, \varphi)$ is an edge in σ, this completes the proof. $\square$

EXAMPLE 5.31. A consequence of the lemma above is that any cycle in G_L contains a K-edge $(K\varphi, \varphi)$, where the end node φ is ungrounded. The cycle in Figure 5.5 contains the K-edge

$$\big(\psi, \forall x \neg \left(\mathit{About}\,(x, \mathit{striped}) \wedge Kx\right)\big),$$

where the end node is an ungrounded sentence. The cycle also contains another K-edge $(K\psi, \psi)$, but in this edge the end node is the grounded sentence

$$K\forall x \neg \left(\mathit{About}\,(x, \mathit{striped}) \wedge Kx\right).$$

From Lemma 5.30 we get important information on where the regular sentences appear in G_L. More precisely, we have the following result.

LEMMA 5.32. *If φ is a regular sentence in L, then φ is not contained in any infinite paths in G_L.*

PROOF. Let φ be an arbitrary regular sentence in L. Assume to obtain a contradiction that φ is the first node in an infinite path σ in G_L.

CLAIM. Every sentence in σ is regular.

PROOF OF CLAIM. It suffices to prove that if (r_1, r_2) is an edge in σ and r_1 is regular, then r_2 is regular as well. So let an arbitrary edge (r_1, r_2) in σ be given and assume that r_1 is regular. If (r_1, r_2) is a $\wedge$-, $\neg$- or K-edge, then it immediately follows that r_2 must be regular as well, by the definition of the regular formulas (Definition 4.9). The only case left to check is if (r_1, r_2) is a $\forall$-edge. In that case we have $r_1 = \forall x \varphi(x)$ and $r_2 = \varphi(\tau)$ for some formula φ and term τ. Since r_1 is regular, the formula $\varphi(x)$ can not contain Kx. But then the regularity of $\varphi(\tau)$ follows directly from the regularity of $\varphi(x)$. This completes the proof of the claim.

The claim contradicts Lemma 5.30, since by that lemma every infinite path contains a K-edge where the end node is ungrounded. We have, however, just constructed an infinite path σ in which every sentence is regular and thus grounded. □

The above lemma gives us some information about the self-referential sentences. Suppose φ is an indirectly self-referential sentence in U_L. Then, by definition, φ is contained in a cycle in G_L. It follows that φ must also be contained in an infinite path in G_L. By the lemma above, the sentence φ can then not be regular. This argument shows that any indirectly self-referential sentence in U_L is non-regular. This fact should be compared with the Rivières-Levesque theorem (Theorem 4.10). The theorem proves that we can consistently instantiate the reflection principles A1–A4 and A2–A6 with the regular sentences. By the argument just given, none of these sentences are (indirectly) self-referential. Thus by only instantiating the reflection principles over the regular sentences we effectively avoid instantiating over any self-referential ones. This explains how restricting instantiation to the regular sentences can give a consistent theory. By only including the regular sentences in our instantiation class, we exclude all self-referential sentences from this class, including the formal knower sentences causing the inconsistency.

As mentioned, it is our goal to find a larger and more useful set of consistent instances of the reflection principles than the one provided by des Rivières and Levesque. The results proven above are going to be very helpful in reaching this goal. Before we can apply these results, however, we need to show how questions of consistency of first-order agent theories relate to properties of G_L. This is the subject of the following section.

5.4. From Sentence Nets to Consistent Reflection Principles

We will now show how the question of consistency of reflection principles can be reduced to the question of non-paradoxicality of certain dependency graphs.

Definition 5.33. *Let M be a set of sentences in L. We define U_M to be the sentence net obtained from U_L by removing every clause $s_{K\varphi} : s_\varphi$ for which φ is not in M. The dependency graph of U_M is denoted G_M.*

The following lemma is a trivial consequence of the definition.

Lemma 5.34. *Let M be as above. The graph G_M is the subgraph of G_L obtained by removing all K-edges $(K\varphi, \varphi)$, where φ is not in M.*

Example 5.35. Consider the graph $G_\emptyset$, that is, the graph G_M where $M = \emptyset$. This graph is obtained from G_L by removing all K-edges. By Lemma 5.21, this graph can not contain any infinite paths and thus no cycles. It is a well-founded graph. The non-wellfounded subgraphs of G_L presented in figures 5.4, 5.5 and 5.6 all become subgraphs of $G_\emptyset$ when we remove all the K-edges. We immediately see that removing these edges make the subgraphs well-founded. By removing the K-edges, all the cycles will be "cut through".

The purpose of defining the reduced sentence nets U_M is illustrated by the following two results. The first result relates interpretations of L to interpretations of U_M. The second result relates consistency of reflection principles to non-paradoxicality of U_M (and, indirectly, G_M).

Lemma 5.36. *Let M be a set of sentences in L. If U_M has a model I, then there exists a Herbrand interpretation J of L satisfying*

(i) *For every sentence φ in L,*

$$I(s_\varphi) = t \;\Leftrightarrow\; J \models \varphi. \tag{5.9}$$

(ii) *J is a model of Q.*

(iii) *$J \models K\varphi \leftrightarrow \varphi$, for all φ in M.*

Proof. Assume U_M has a model I. Let J be the Herbrand interpretation of L having the following set of true atomic sentences

$$\{\varphi \in L \mid \varphi \text{ is an atomic sentence and } I(s_\varphi) = t\}\,.$$

Lemma 3.14 guarantees that a unique such Herbrand interpretation exists. We now prove (i) by induction on the syntactic complexity of φ. For the base case, assume φ is an atomic sentence in L. Then (5.9) holds as a trivial consequence of the way we defined J. This proves the base case. To prove (5.9) for sentences φ on the form $\neg\alpha$, we use the fact that U_M contains the clause $s_{\neg\alpha} : \neg s_\alpha$. Since I is a model of U_M, this clause implies that we have

$$I(s_{\neg\alpha}) = I(\neg s_\alpha).$$

We now get

$$I(s_{\neg\alpha}) = t \;\Leftrightarrow\; I(\neg s_\alpha) = t \;\Leftrightarrow\; I(s_\alpha) = f \;\overset{\text{i.h.}}{\Leftrightarrow}\; J \not\models \alpha \;\Leftrightarrow\; J \models \neg\alpha,$$

where the second to last equivalence is by induction hypothesis. This proves (5.9) for all sentences φ on the form $\neg\alpha$. When φ is on the form $\forall x\alpha(x)$, we can use the fact that U_M contains the clause

$$s_{\forall x\alpha(x)} : \bigwedge_{\tau\in\text{Terms}(L)} s_{\alpha(\tau)} \tag{5.10}$$

to prove that

$$\begin{aligned}
I(s_{\forall x\alpha(x)}) = t \ &\overset{(5.10)}{\Leftrightarrow}\ I(\textstyle\bigwedge_{\tau\in\text{Terms}(L)} s_{\alpha(\tau)}) = t \\
&\Leftrightarrow\ I(s_{\alpha(\tau)}) = t \text{ for all } \tau \in \text{Terms}(L) \\
&\overset{\text{i.h.}}{\Leftrightarrow}\ J \models \alpha(\tau) \text{ for all } \tau \in \text{Terms}(L) \\
&\Leftrightarrow\ J \models \forall x\alpha(x).
\end{aligned}$$

The last equivalence above follows from the fact that J is a Herbrand model. The chain of equivalences above prove (5.9) in the case where φ is on the form $\forall x\alpha(x)$. The final case where φ is on the form $\alpha \wedge \beta$ is proved similarly to the two previous cases. Thus our induction proof of (i) is complete.

To prove (ii), it suffices to show that for every atomic sentence φ in L we have

$$Q \vdash \varphi \ \Rightarrow\ J \models \varphi. \tag{5.11}$$

$$Q \vdash \neg\varphi \ \Rightarrow\ J \models \neg\varphi. \tag{5.12}$$

Using (i) and the fact that I is an interpretation of U_M, we get

$$Q \vdash \varphi \ \Rightarrow\ s_\varphi : \textit{true} \in U_M \ \Rightarrow\ I(s_\varphi) = I(\textit{true}) = t \ \overset{\text{(i)}}{\Rightarrow}\ J \models \varphi.$$

This shows the implication (5.11). The implication (5.12) is proved by

$$Q \vdash \neg\varphi \ \Rightarrow\ s_\varphi : \textit{false} \in U_M \ \Rightarrow\ I(s_\varphi) = I(\textit{false}) = f \ \overset{\text{(i)}}{\Rightarrow}\ J \not\models \varphi \ \Rightarrow\ J \models \neg\varphi.$$

Hereby (ii) is proved.

To prove (iii), let an arbitrary sentence φ in M be given. Then U_M contains the clause $s_{K\varphi} : s_\varphi$. This implies that

$$I(s_{K\varphi}) = I(s_\varphi). \tag{5.13}$$

Using this equality and (i) we now get

$$J \models K\varphi \ \overset{\text{(i)}}{\Leftrightarrow}\ I(s_{K\varphi}) = t \ \overset{(5.13)}{\Leftrightarrow}\ I(s_\varphi) = t \ \overset{\text{(i)}}{\Leftrightarrow}\ J \models \varphi.$$

This proves $J \models K\varphi \leftrightarrow \varphi$, and the proof is hereby complete. $\square$

Theorem 5.37. *Let M be a set of sentences in L closed under $\wedge$, $\neg$ and K.*[4] *Suppose S is the theory Q extended with a set of reflection principles instantiated over M. If U_M has a model then S is consistent.*

Proof. Assume U_M has a model. Then by the lemma above, there exists a Herbrand interpretation J which is a model of Q and in which $K\varphi \leftrightarrow \varphi$ holds for all φ in M. By Lemma 3.16, this implies that all of the reflection principles A1–A7 instantiated over M hold in J. The interpretation J is therefore a model of S, and this shows S to be consistent. $\square$

The above theorem is very important. It shows that proving consistency results of the Rivières-Levesque and Morreau-Kraus type can be reduced to the problem of finding sets M for which U_M has a model. This can be further reduced to the problem of finding sets M for which G_M is non-paradoxical, since if G_M is non-paradoxical then U_M has a model, by definition. That is, the consistency problem for our reflection principles becomes reduced to proving properties of certain graphs. In the next chapter we will use this to prove two consistency results generalising the Rivières-Levesque theorem.

5.5. Relations to Logic Programming

The reader familiar with logic programming will probably have noticed a considerable similarity between, on the one hand, sentence nets and their dependency graphs, and, on the other hand, logic programs and their dependency graphs. We will now explore these relationships.

Our sentence nets are most closely related to *propositional logic programs* (*ground logic programs*). These can be defined as follows. We begin with a non-empty set of propositional letters called **atoms**. We take the special symbols *true* and *false* to be among these atoms. A **literal** is either an atom A or its negation $\neg A$. A (**program**) **clause** is an expression of the form

$$H \leftarrow L_1, \dots, L_n$$

where H is an atom and all L_i are literals. The atom H is called the **head** of the clause and $L_1, \dots, L_n$ the **body**. Program clauses can also be defined simply to be formulas of propositional logic on the form

$$H \leftarrow L_1 \wedge \cdots \wedge L_n,$$

where H and L_i are as above. A (**propositional**) **program** is defined to be a set of clauses (possibly an infinite set). Non-propositional programs—that is,

[4]By *closed under* $\wedge$, $\neg$ and K we mean that if φ and ψ are sentences in M, then so are $\varphi \wedge \psi$, $\neg\varphi$ and $K\varphi$.

programs where the atoms are atomic formulas of first-order predicate logic—can be translated into propositional programs by **grounding** them: replacing every clause by the set of *ground instances* (closed instances) of it.

Clauses of propositional programs can be translated into sentence net clauses in the following simple way. Let

$$H \leftarrow L_1 \wedge \cdots \wedge L_n$$

be any program clause. The **translation** of the clause is defined to be the sentence net clause given by

$$H : \bigwedge \{L_1, \ldots, L_n\} .$$

This translation does not immediately give us a way to translate programs into sentence nets, however, since in programs we can have several clauses with the same head as in

$$\begin{aligned} &A \leftarrow \neg B, C \\ &A \leftarrow B \\ &C \leftarrow \neg B. \end{aligned} \tag{5.14}$$

In sentence nets, no two clauses can have the same head. Sentence nets are more directly related to the *Clark completions* of propositional programs. The *Clark completion* of a propositional program P is defined as the program $\mathrm{comp}(P)$ containing the following equivalences of infinitary propositional logic: For each atom A in P,

- If A does not appear as head of any clause in P, then $A \leftrightarrow \mathit{false} \in \mathrm{comp}(P)$.
- Otherwise we have

  $$A \leftrightarrow \bigvee_{i \in I} \left(L_1^i \wedge \cdots \wedge L_{n_i}^i\right) \in \mathrm{comp}(P),$$

  where $\{A \leftarrow L_1^i \wedge \cdots L_{n_i}^i \mid i \in I\}$ is the set of clauses in P with head A.

In the Clark completion of a program, every formula is on the form

$$A \leftrightarrow \varphi,$$

where φ is a possibly infinitary propositional formula. We again call A the *head* and φ the *body*. In the Clark completion, every atom occurs as the head of exactly one formula. The Clark completion of a program can therefore immediately be translated into a sentence net by simply replacing the '$\leftrightarrow$' in every formula by ':'. It is easy to see that the sentence net obtained in this way has a model if and only if the Clark completed program has a model.

Any propositional program P can be assigned a **dependency graph** as follows. The dependency graph is a directed graph with *signed edges*, that is, it contains two different types of edges called *positive* and *negative edges*. The nodes of the graph are the atoms occurring in the program P. There is a *positive* edge from the node A to the node B if and only if there is a clause $A \leftarrow L_1 \wedge \cdots L_n$ in P such that $L_i = B$ for some $1 \leq i \leq n$. If $L_i = \neg B$ then there is a *negative* edge from A to B. The dependency graph of the program (5.14) looks as follows

$$\begin{tikzcd} A \arrow[rr, bend left, "+"] \arrow[rr, bend right, "-"'] \arrow[dr, "+"'] & & B \\ & C \arrow[ur, "-"'] & \end{tikzcd}$$

The only significant difference between these dependency graphs and the ones we have defined for sentence nets is that our graphs are *unsigned*. We can think of this difference in the following way. Whereas the dependency graphs of programs contain information about the way in which atoms semantically depend on one another (positively or negatively), our dependency graphs only contain information about the *patterns* of semantic dependency.

For the material in this chapter, we could easily have used Clark completed propositional programs instead of sentence nets. This would have given us a number of the basic definitions and results almost for free, since they correspond more or less directly to well-known definitions and results within logic programming. This concerns definitions 5.4, 5.8, 6.1, 6.2 and 6.6, lemmata 6.3, 6.5 and 6.7, and Theorem 6.8. Of these, we would still have been required to state definitions 6.2 and 6.6 as well as lemmata 6.3 and 6.5 to avoid the need of distinguishing between programs and their Clark completions.

When considering our definitions and results concerning the relation of sentence nets to agent theories, the situation is somewhat different. None of these definitions and results would gain from being put into the framework of propositional logic programs (but we would not loose anything essential by doing it either).

5.6. Chapter Notes

Mathematical objects similar to our sentence nets have been considered in many different places in the literature. Our sentence nets are most directly connected to the *language of paradox* introduced by Cook [2002]. The formal language defined in Definition 5.1 is an extension of this language. Cook uses his language to study various semantic paradoxes, in particular infinite ones related to Yablo's paradox [Yablo, 1985; 1993]. He also associates dependency graphs (*dependency relations*) with sets of sentences, and relates questions of paradoxicality of such sets to graph theoretical properties of the associated dependency graphs. However, our framework is considerably more general than his. Sandy Hodges has recently been considering similar sets of mutually referring sentences, but in a less formal framework. We have borrowed the term *sentence net* from him, since he used this term when presenting his ideas to the Foundations of Mathematics (FOM) discussion group. Visser [1989] considers *stipulation lists*, which are functions from a set of propositional symbols into propositional sentences over these symbols. They correspond to sentence nets in which only finite disjunctions and conjunctions are used. Finally, we should mention the *theory of definitions* by Gupta and Belnap [1993], which also contains many elements closely related to our sentence nets and dependency graphs. Even though there are many mathematical objects introduced in the literature that are more or less similar to our sentence nets, we did not find any that we could have used for our purpose without modifications. Most of the material on sentence nets and dependency graphs in sections 5.1 and 5.2 appear in Bolander [2003b].

All the material in sections 5.3 and 5.4 is due to the author. In these sections we make the important connection between agent theories and dependency graphs, and show how the consistency problem for reflection principles is related to properties of certain dependency graphs. Some of this material appears in Bolander [2003a].

The definition of the *K-depth* of a formula (Definition 5.23) is adapted from Lakemeyer [1992]. He defines a similar notion—which he calls *depth of a formula*—for formulas of first-order modal logic.

CHAPTER 6

Consistency Results for Agent Theories

We will now apply the methods developed in the previous chapter to prove our main results concerning the consistent treatments of introspective reasoning. Our two main results are Theorem 6.9 and Theorem 6.23. They both generalise the Rivières-Levesque theorem. We will be giving examples of how our results can be applied in constructing agents with considerably stronger introspective abilities than those the Rivières-Levesque theorem gives rise to.

6.1. First Strengthened Consistency Result

Below we will prove that all well-founded graphs are non-paradoxical. Together with Theorem 5.37, this shows that if M is a set of sentences for which G_M is well-founded, then we can consistently instantiate our reflection principles over M. From this we will then be able to prove that the reflection principles can consistently be instantiated over the grounded sentences. This is a generalisation of the Rivières-Levesque theorem.

6.1.1. The Result. To prove our result generalising the Rivières-Levesque theorem, we need to introduce a few new notions and a little extra machinery. All of this machinery is fairly standard material within the areas of logic programming semantics and formal theories of truth.

Definition 6.1. *Let U be a sentence net over a set N. A **partial interpretation** of U is a map I from N to $\{t, f, \perp\}$, where $\perp$ denotes "undefined". Partial interpretations are extended to give values to the expressions* true *and* false *by letting $I(\text{true}) = t$ and $I(\text{false}) = f$. Partial interpretations extend further to give values to all compound expressions in $\mathscr{L}_N$ by using the strong Kleene valuation schemes for $\neg$, $\bigvee$ and $\bigwedge$.*[1] *The **domain** of I, denoted* $\text{dom}(I)$, *is the set $\{s \in N \mid I(s) \neq \perp\}$. We say that a sentence s is **defined***

[1]The *strong Kleene valuation schemes* are defined in Kleene [1964]. In our case they amount to the following conditions for the connectives $\neg$ and $\bigwedge$.

$$I(\neg E) = \begin{cases} t, & \text{if } I(E) = f \\ f, & \text{if } I(E) = t \\ \perp, & \text{if } I(E) = \perp \end{cases} \qquad I(\textstyle\bigwedge_{i \in M} E_i) = \begin{cases} t, & \text{if } I(E_i) = t \text{ for all } i \in M \\ f, & \text{if } I(E_i) = f \text{ for some } i \in M \\ \perp, & \text{otherwise.} \end{cases}$$

*in I if $s \in \mathrm{dom}(I)$ and **undefined** otherwise. A **total interpretation** of U is a partial interpretation whose domain is N.*

We can think of a sentence net as giving us a kind of "update operator" or "revision operator". If a sentence net contains a clause $s\!:\!E$ and we know E to be true, then we should update our interpretation to let s be true as well. The following operator, Φ_U, is intended to capture one pass of such an update (this is completely analogous to what Fitting does in [Fitting, 2002a]).

Definition 6.2. *Let U be a sentence net. An associated mapping Φ_U, from partial interpretations to partial interpretations, is defined as follows. For every partial interpretation I of U, $\Phi_U(I)$ is the partial interpretation defined by*

$$\Phi_U(I)(s) = \begin{cases} I(E), & \textit{if } s\!:\!E \textit{ is a clause in } U. \\ I(s), & \textit{if there is no clause with head } s \textit{ in } U. \end{cases}$$

Lemma 6.3. *Let U be a sentence net. If I is a total interpretation of U and a fixed point of Φ_U, then I is a model of U.*

Proof. Assume I is a total interpretation of U and a fixed point of Φ_U. To prove that I is a model of U, we only have to show that $I(s) = I(E)$ holds for all clauses $s\!:\!E$ in U. Let thus $s\!:\!E$ be an arbitrary clause in U. Since I is a fixed point of Φ_U, we have $\Phi_U(I) = I$. This implies

$$I(s) = \Phi_U(I)(s) = I(E),$$

as required. □

The set $\{t, f, \bot\}$ can be given a partial ordering $\sqsubset$ by letting $\bot \sqsubset f$ and $\bot \sqsubset t$, with $x \sqsubset y$ not holding in any other cases. We can then define $\sqsubseteq$ by

$$x \sqsubseteq y \Leftrightarrow x = y \text{ or } x \sqsubset y.$$

The ordering $\sqsubseteq$ can be extended to partial interpretations of a sentence net U in a point-wise fashion: $I \sqsubseteq J$ if and only if $I(s) \sqsubseteq J(s)$ for all sentences s in U. When $I \sqsubseteq J$ we say that J **extends** I. We call a set X of partial interpretations **consistent** if for every pair I, I' in X there is a partial interpretation J extending both I and I'. We have the following well-known results concerning the ordering $\sqsubseteq$.

Lemma 6.4 (After Visser [1989]). *Suppose U is a sentence net and X is the set of partial interpretations of U. Then the following conditions hold.*

(i) *Every consistent subset Y of X has a least upper bound with respect to $\sqsubseteq$. We denote this least upper bound by $\sqcup Y$.*

The condition for the connective $\bigvee$ is obtained by interchanging "all" and "some" in the condition for $\bigwedge$.

(ii) *For any map* $g : X \to X$ *which is monotonic with respect to* $\sqsubseteq$*, the set* $\{I \in X \mid I \sqsubseteq g(I)\}$ *has a maximal element* I^**. The partial interpretation* I^* *is a fixed point of* g*.*
(iii) *Suppose* $g : X \to X$ *is monotonic with respect to the ordering* $\sqsubseteq$*, and* I *is an element in* X *satisfying* $I \sqsubseteq g(I)$*. Then* g *has a least fixed point extending* I*.*

Proof. We only have to prove that $(X, \sqsubseteq)$ is a *coherent complete partial order*, since in Section 2 of Visser [1989] the properties (i)–(iii) are proven to hold for any such order. A coherent complete partial order (henceforth *ccpo*) is a partial order $(D, \leq)$ for which every consistent subset of D has a least upper bound in D. It is trivial to check that $(\{t, f, \bot\}, \sqsubseteq)$ is a ccpo. By item (iii) of Lemma 2.7 in Visser [1989], it then follows that the set $(X, \sqsubseteq)$ is a ccpo as well. This completes the proof. □

Lemma 6.5. *The update operator* Φ_U *is monotonic with respect to the ordering* $\sqsubseteq$*.*

Proof. Assume $I \sqsubseteq J$. We have to prove that $\Phi_U(I) \sqsubseteq \Phi_U(J)$. To do this, let s be an arbitrary sentence in U. Then we have to prove

$$\Phi_U(I)(s) \sqsubseteq \Phi_U(J)(s). \tag{6.1}$$

We first consider the case where s is the head of some clause $s : E$ in U. In this case we have

$$\Phi_U(I)(s) = I(E) \sqsubseteq J(E) = \Phi_U(J)(s),$$

where $I(E) \sqsubseteq J(E)$ follows from the assumption $I \sqsubseteq J$ and the monotonicity wrt. $\sqsubseteq$ of the strong Kleene valuations used to assign values to compound expressions (cf. Definition 6.1). This proves (6.1) in the case where s is the head of a clause in U. If s is not the head of any clause, we get

$$\Phi_U(I)(s) = I(s) \sqsubseteq J(s) = \Phi_U(J)(s).$$

Thus (6.1) holds in both cases, and the proof is therefore complete. □

Suppose U is a sentence net and let I be the partial interpretation of U given by

$$I(s) = \begin{cases} f, & \text{if there is no clause in } U \text{ with head } s. \\ \bot, & \text{otherwise.} \end{cases} \tag{6.2}$$

We will now show that $\Phi_U(I)$ extends I, that is, $I \sqsubseteq \Phi_U(I)$. To do this, let s be any sentence in U. We have to prove $I(s) \sqsubseteq \Phi_U(I)(s)$. If s is the head of a clause in U we have $I(s) = \bot$, and thus $I(s) \sqsubseteq \Phi_U(I)(s)$ holds trivially. If s is not the head of a clause, then by definition of Φ_U we have $\Phi_U(I)(s) = I(s)$, so again the inequality $I(s) \sqsubseteq \Phi_U(I)(s)$ holds. It now follows from Lemma

6.5 and item (iii) of Lemma 6.4 that Φ_U has a least fixed point extending I. This fixed point is quite important, so it deserves to be given a name.

Definition 6.6. *Let U be a sentence net and let I be the partial interpretation of U given by (6.2) above. The least fixed point of Φ_U extending I is called the* ***minimal interpretation*** *of U.*

The minimal interpretation of a sentence net U corresponds closely to Kripke's least fixed point in Kripke [1975]. The minimal interpretation has the following important property.

Lemma 6.7. *Let U be a sentence net and let G denote its dependency graph. If a node s in G is not contained in any infinite path, then it is defined in the minimal interpretation of U.*

Proof. Let I denote the partial interpretation given by (6.2) above and let I_0 denote the minimal interpretation of U. By definition of I_0, we have $I \sqsubseteq I_0$. Let M be the set of nodes in G not contained in any infinite paths. Let M' be the subset of M consisting of the nodes undefined in I_0, that is, the nodes $s \in M$ with $I_0(s) = \bot$. We have to show that M' is the empty set. Assume to obtain a contradiction that M' is non-empty. Then there must exist a sentence s in M' for which all sentences in $\Gamma_G(s)$ are not in M'. This fact is realised by noting that otherwise we could from any sentence in M' construct an infinite path consisting only of elements in M', contradicting that M' is a subset of M. The sentence s must be the head of a clause in U, since otherwise the definition of I and the fact that I_0 extends I would give us

$$I_0(s) \sqsupseteq I(s) = f,$$

contradicting that s is an element of M'. Let thus $s : E$ be the clause with head s in U. Then $\Gamma_G(s)$ is the set of sentences occurring in E. By choice of s, we have $\Gamma_G(s) \subseteq M - M'$. This means that all elements of $\Gamma_G(s)$ are defined in I_0. The expression E must therefore also be defined in I_0. That is, $I_0(E) \neq \bot$. Using that I_0 is a fixed point of Φ_U, we now finally get

$$I_0(s) = \Phi_U(I_0)(s) = I_0(E) \neq \bot.$$

This contradicts, once again, the choice of s. The proof is hereby complete. □

Lemma 6.3 and Lemma 6.7 now give us the following promised result.

Theorem 6.8. *Suppose G is a well-founded graph. Then G is non-paradoxical.*

Proof. Let U be any sentence net with dependency graph G. We have to show that U has a model. Let I denote the minimal interpretation of U. We want to show that I is a model of U. Note that since G is well-founded, it does not contain any infinite paths, and I must therefore be defined on every sentence in U. This follows from Lemma 6.7. Now Lemma 6.3 immediately gives us that I is a model of U. □

We are now finally ready to prove our first consistency result strengthening the Rivières-Levesque theorem. Recall that we have decided to take L to denote a *fixed* first-order agent language.

THEOREM 6.9 (First main result). *Suppose S is the theory Q extended with a set of reflection principles instantiated over the grounded sentences of L. Then S is consistent.*

PROOF. Using Theorem 5.37, it is sufficient to prove that U_M has a model when M is the set of grounded sentences. This can be proved by showing that G_M is non-paradoxical. We will prove that G_M is well-founded. Then the required conclusion follows from Theorem 6.8. Assume to obtain a contradiction that G_M is not well-founded. Then it must contain an infinite path. By Lemma 5.30, this path must contain a K-edge $(K\varphi, \varphi)$ where φ is ungrounded (we are using that G_M is a subgraph of G_L). But this immediately contradicts Lemma 5.34, and the proof is thus complete. □

In the following section we will look at some of the consequences of Theorem 6.9 concerning consistent treatments of agent introspection.

6.1.2. Discussion. We have already noted that all regular sentences are grounded (Lemma 5.27), so Theorem 6.9 has the Rivières-Levesque theorem as an immediate corollary. The question is: How much more general than the Rivières-Levesque theorem is our result? First of all, it is more general in concerning *all* combinations of the reflection principles and not only the two subsets A1–A4 and A2–A6. However, it is implicit in the construction of des Rivières and Levesque that their result can also be generalised in this way. Furthermore, our result is more general since it covers the larger set of *grounded sentences*. The question is whether these sentences are expressive enough to allow the kind of introspective reasoning we have been aiming for. To answer this question, we take another look at the list of sentences considered in Section 4.3.1:

(i) $K\, On\,(black, floor)$
(ii) $K\neg K\, On\,(striped, white)$
(iii) $K\,(\neg K\, On\,(striped, floor) \wedge \neg K \neg On\,(striped, floor))$
(iv) $K\neg\exists x\,(About\,(x, striped) \wedge Kx)$
(v) $K_{ole}\,\forall x\,(About\,(x, cool\ jazz) \wedge K_{sue}\,x \to K_{bill}\,x)$
(vi) $K_{john}\,\exists x\,(K_{bill}\,x \wedge \neg K_{john}\,x)$
(vii) $\forall x \forall y\,(IterateK\,(\varphi, n, x) \wedge y < n \to K_y\,(x))$

(viii) $\forall x_1 \forall x_2 \, (Neg\,(x_1, x_2) \rightarrow \neg\,(Kx_1 \wedge Kx_2))$

We have already noted that of these sentences only (i)–(iii) are regular. All of the sentences (i)–(vi) are grounded, however, since none of these contain Kx as a subformula for any variable x (they only contain Kx in coded form). This means that the theorem above gives a significant improvement over the Rivières-Levesque theorem. Our result shows that we can also consistently allow agents to reason about sentences such as (iv)-(vi). The two remaining sentences, (vii) and (viii), are not grounded, so the theorem does not guarantee that such sentences can be reasoned about consistently. Neither does it guarantee that we can reason about the sentences obtained from removing the initial K from (iv)-(vi), that is, the sentences

(iv') $\neg \exists x \, (About\,(x, striped) \wedge Kx)$
(v') $\forall x \, (About\,(x, cool\ jazz) \wedge K_{sue}\, x \rightarrow K_{bill}\, x)$
(vi') $\exists x \, (K_{bill}\, x \wedge \neg K_{john}\, x)$

We would like our agents to be able to reason about such sentences as well, if this can be done consistently. We must therefore look for a way to strengthen the result above.

Using Theorem 5.37, the result above can be strengthened if we can find a set M larger than the set of grounded sentences for which G_M is still non-paradoxical. Unfortunately, no such set exists! It is possible to show that M only needs to contain a single ungrounded sentence for G_M to become paradoxical. We will not give a proof of this statement here, but simply note that it can be proven by showing the following two facts

- Any cyclic graph is paradoxical.
- If M contains an ungrounded sentence, then G_M is cyclic.

This implies that we are in need of an alternative strategy if we wish to strengthen the result above.

An obvious possible strategy would be to see if there, given M, are any edges in G_M which are "dispensable". If it is somehow possible to thin out the graph without destroying its essential properties, then we might be able to show the well-foundedness of the modified graph, even if M contains ungrounded sentences. In the following section we pursue this strategy.

6.2. Second Strengthened Consistency Result

We start this section by giving an example demonstrating the intuition behind the strategy of "thinning out" (reducing) dependency graphs.

Example 6.10 (The *About* predicate)**.** In Example 4.1, we showed that our formalisation of the *About* predicate suffers from problems related to self-reference. We demonstrated that an agent can become inconsistent simply from reasoning about a sentence expressing: "I do not know anything about the striped block". The problem is that this sentence itself constitutes a piece of knowledge about the striped block.

The problems can be avoided by doing things slightly different. Let us agree to call a formula φ in L **objective** if it does not contain any occurrence of K, that is, if it is a formula in $L - \{K\}$ (Lakemeyer [1992] defines objective formulas in a similar way for a corresponding language of first-order modal logic). When an agent expresses that nothing is known about an object c, then in most cases the agent probably intends to be saying: "nothing *objective* is known about c". Let Obj be the set of Gödel numbers of objective sentences in L. Since Obj is trivially recursive, there exists a formula $Obj(x_1)$ representing Obj in any agent theory. To express that nothing objective is known about the striped block, we can then simply write

$$\forall x \left(About\left(x, striped\right) \wedge Obj\left(x\right) \rightarrow \neg Kx \right). \tag{6.3}$$

Let us denote this formula φ. Since φ is not objective, it will not itself be one of the sentences it claims not to be known. This will block the reasoning reasoning carried out in Example 4.1, where the sentence corresponding to (6.3) was instantiated with (the Gödel code of) itself. If we instantiate φ with the Gödel code of φ, we get

$$About\left(\varphi, striped\right) \wedge Obj\left(\varphi\right) \rightarrow \neg K\varphi.$$

This sentence is simply a theorem in Q, since it follows from our definition of $Obj(x_1)$ that

$$Q \vdash \neg Obj\left(\varphi\right).$$

The sentence will therefore not be able to do any harm.

There is, however, still a problem with *About*. Consider the following sentence

$$On\left(striped, floor\right) \vee \neg On\left(striped, floor\right).$$

This sentence is valid in first-order predicate logic, so if the agent's knowledge base contains reflection principle A3, then the agent will be able to infer

$$K\left(On\left(striped, floor\right) \vee \neg On\left(striped, floor\right)\right). \tag{6.4}$$

At the same time it will, from φ, be able to infer

$$\neg K\left(On\left(striped, floor\right) \vee \neg On\left(striped, floor\right)\right), \tag{6.5}$$

since $On(striped, floor) \vee \neg On(striped, floor)$ is an objective sentence containing an occurrence of *striped*. Thus, the agent will be able to infer the contradiction obtained from taking the conjunction of (6.4) and (6.5). This problem does not apply to the situation considered in Example 4.1, however, since the only reflection principle we included there was A1. To avoid contradictions such as the one we just inferred, we must make some further restrictions to the way we treat *About*. Let $Lit(x_1)$ be a formula representing the set of *closed literals* in L, that is, the set of atomic and negated atomic sentences. Then we can replace φ by

$$\psi \stackrel{df}{=} \forall x \left(About(x, striped) \wedge Obj(x) \wedge Lit(x) \rightarrow \neg Kx \right). \tag{6.6}$$

The sentence ψ expresses that no objective literals concerning *striped* are known. If ψ is contained in an agent's knowledge base, the agent will be able to infer

$$\begin{aligned} &\neg K\, On(striped, floor) \\ &\neg K \neg On(striped, floor) \\ &\neg K\, On(black, striped) \\ &\quad \vdots \end{aligned}$$

but not problematic ones such as

$$\begin{aligned} &\neg K \left(On(striped, floor) \vee \neg On(striped, floor) \right) \\ &\neg K \psi. \end{aligned}$$

To be certain that there are no further problems in our formalisation of *About*, we would like to prove that sentences such as ψ can consistently be reasoned about, that is, we can consistently instantiate our reflection principles with such sentences. The strongest consistency result at our disposal concerning instances of the reflection principles is Theorem 6.9. This theorem does unfortunately not guarantee us that a sentence such as ψ can consistently be instantiated with, since ψ is not grounded. We do still not believe that ψ is paradoxical, however, since ψ does not refer to itself in the same way as the sentence considered in Example 4.1 did. The occurrence of $\neg Kx$ in ψ is *protected* by the formula $Obj(x)$, which ensures that only the non-knowledge of a set of *objective* sentences is expressed. Since ψ is not objective, there is no self-reference involved, and thus everything should be fine. With the way we have defined our dependency graphs, these graphs are however not differentiated enough to allow us to see that ψ is not self-referential. Consider the subgraph of G_L presented in Figure 6.1. We have here replaced the ψ from (6.6) by the sentence that it abbreviates, which is

$$\forall x \neg \left(About(x, striped) \wedge Obj(x) \wedge Lit(x) \wedge Kx \right).$$

We see that ψ is (indirectly) self-referential in U_L, since it is contained in a cycle. Since ψ is not objective, we have

$$Q \vdash \neg Obj(\psi).$$

$$\begin{tikzcd}
\forall x \neg \left(\mathit{About}\,(x, \mathit{striped}) \wedge \mathit{Obj}\,(x) \wedge \mathit{Lit}\,(x) \wedge Kx \right) \arrow[d, "\forall"] & \\
\neg \left(\mathit{About}\,(\psi, \mathit{striped}) \wedge \mathit{Obj}\,(\psi) \wedge \mathit{Lit}\,(\psi) \wedge K\psi \right) \arrow[d, "\neg"] & \\
\mathit{About}\,(\psi, \mathit{striped}) \wedge \mathit{Obj}\,(\psi) \wedge \mathit{Lit}\,(\psi) \wedge K\psi \arrow[d, "\wedge"] \arrow[dr, "\wedge"] & \\
\mathit{About}\,(\psi, \mathit{striped}) \wedge \mathit{Obj}\,(\psi) \wedge \mathit{Lit}\,(\psi) & K\psi \arrow[uuul, bend right, "K"']
\end{tikzcd}$$

FIGURE 6.1. A subgraph of G_L.

From this alone it follows that

$$Q \vdash \neg \left(\mathit{About}\,(\psi) \wedge \mathit{Obj}\,(\psi) \wedge \mathit{Lit}\,(\psi) \wedge K\psi \right).$$

Thus, in this case, we do not need all of the semantic dependencies expressed by the edges of the graph to determine the semantic value of ψ. In particular, we can drop the edge leading from $\mathit{About}\,(\psi, \mathit{striped}) \wedge \mathit{Obj}\,(\psi) \wedge \mathit{Lit}\,(\psi) \wedge K\psi$ to $K\psi$, since the semantic value of $K\psi$ is not needed to determine the semantic value of the conjunction. If we can somehow *reduce* our graphs by dropping edges that express semantic dependencies which are not used, then we might be able to correctly see that ψ is actually not self-referential; and that such sentences can also be treated consistently. We are now going to show how such reductions of graphs can be carried out.

6.2.1. The Result. The following definition gives us a way to reduce sentence nets and thus dependency graphs.

DEFINITION 6.11. *Let U be a sentence net and let I be a partial interpretation of U. By the* ***reduction*** *of U* ***modulo*** *I we understand the sentence net obtained from U by performing the following two transformations*

- *Remove every clause with head s for which $I(s) = t$ or $I(s) = f$.*
- *Replace in the remaining clauses every occurrence of a sentence s for which $I(s) = t$ with the expression* true *and every occurrence of an s for which $I(s) = f$ by* false*.*

The following lemma is a trivial consequence of the way we have defined reductions of sentence nets.

LEMMA 6.12. *Let U and I be as above. Let G denote the dependency graph of U and G' the dependency graph of the reduction of U modulo I. The* *graph G' is obtained from G by removing every edge (r_1, r_2) for which r_1 is defined in I.*

Definition 6.13. *Let U be a sentence net and let I denote the minimal interpretation of U. The reduction of U modulo I is called the* **minimal reduction** *of U. The minimal reduction of U is denoted U^- and its dependency graph G^-.*

The following two results should not come as a surprise.

Lemma 6.14. *Let U be a sentence net. If U^- has a model, then U has a model.*

Proof. Assume that a sentence net U over N is given. Assume further that the minimal reduction U^- of U has a model I^-. We have to show that in this case U has a model as well. Let I_0 denote the minimal interpretation of U. Note that by definition, U^- is a sentence net over the set $N - \mathrm{dom}(I_0)$. Thus I^- and I_0 have disjoint domains, and they therefore have a least upper bound $I^- \sqcup I_0$ (cf. Lemma 6.4(i)). Since I^- has domain $N - \mathrm{dom}(I_0)$ and I_0 has domain $\mathrm{dom}(I_0)$, this least upper bound must be a *total interpretation* of U. We claim that $I^- \sqcup I_0$ is a model of U. To prove this claim, let $s : E$ be any clause in U. We have to prove that

$$I^- \sqcup I_0(s) = I^- \sqcup I_0(E).$$

Assume first that s is defined in I_0, that is, $I_0(s) = t$ or $I_0(s) = f$. Then, since I_0 is a fixed point of Φ_U, we get

$$I^- \sqcup I_0(s) = I_0(s) = \Phi_U(I_0)(s) = I_0(E) = I^- \sqcup I_0(E),$$

as required. Assume then that s is *not* defined in I_0, that is, $s \in N - \mathrm{dom}(I_0)$. Then U^- contains a clause $s : E'$, where E' is obtained from E by replacing every sentence s' for which $I_0(s') = t$ with *true* and every sentence s' for which $I_0(s') = f$ with *false*. From this it immediately follows that

$$I^- \sqcup I_0(E') = I^- \sqcup I_0(E).$$

Since I^- is a model of U^- we then get

$$I^- \sqcup I_0(s) = I^-(s) = I^-(E') = I^- \sqcup I_0(E') = I^- \sqcup I_0(E).$$

This completes the proof. □

Lemma 6.15. *Let φ be any sentence in L. If $Q \vdash \varphi$ then φ does not occur in U_L^- (and thus not in G_L^-).*

Proof. Let φ be given such that $Q \vdash \varphi$. We have to show that s_φ does not occur in the sentence net U_L^-. Let I_0 denote the minimal interpretation of U_L. Then I_0 is a fixed point of the operator Φ_{U_L}. Consider the sentence net $U_\emptyset$. This sentence net is obtained from U_L by removing all clauses of the form $s_{K\varphi}\!:\!s_\varphi$. The dependency graph $G_\emptyset$ of $U_\emptyset$ can therefore not contain any K-edges. By Lemma 5.21, it then follows that there are no infinite paths in $G_\emptyset$. From this we obtain, by Lemma 6.7, that I_0 is a *total* interpretation of $U_\emptyset$. Furthermore, since $U_\emptyset$ is obtained from U_L by removing a set of clauses and since I_0 is a fixed point of Φ_{U_L}, the interpretation I_0 must be a fixed point of $\Phi_{U_\emptyset}$ as well. What we have hereby shown is that I_0 is a total fixed point of $\Phi_{U_\emptyset}$. By Lemma 6.3, I_0 is therefore a model of $U_\emptyset$. It then follows by Lemma 5.36 that there exists a model J of Q satisfying

$$I_0(s_\varphi) = t \Leftrightarrow J \models \varphi. \tag{6.7}$$

Since we have assumed $Q \vdash \varphi$ and since J is a model of Q, we must have $J \models \varphi$ and thus $I_0(s_\varphi) = t$. Since I_0 is the minimal interpretation of U_L, it follows by the definition of U_L^- that every occurrence of s_φ in U_L has been replaced by the expression *true* in U_L^-. In order words, s_φ does not occur in U_L^-, which is the required conclusion. □

We are now ready to define the set of sentences which we are going to prove that our reflection principles can consistently be instantiated with. These sentences are called *protected sentences*. The idea behind them is explained in the following. We know that the ungrounded sentences are the ones causing the problems, since we have already shown it to be safe to instantiate with all grounded sentences (Theorem 6.9). An ungrounded sentence is a sentence containing Kx as a subformula for some variable x. It is thus a sentence containing a subformula of the form $\forall x \varphi(x)$, where $\varphi(x)$ contains Kx. The problem is here that the quantifier $\forall x$ quantifies over all (Gödel codes of) sentences, including the sentence itself. This means that it is a self-referential sentence, and it is thus prone to cause inconsistency. The idea is now to somehow prevent the quantifier $\forall x$ from quantifying over a set of sentences including the sentence in which it itself occurs. In Example 6.10 we hinted at how this can be done. If we for instance write

$$\forall x \left(\mathit{Obj}\,(x) \rightarrow Kx \right),$$

then this will be an ungrounded sentence expressing that all objective sentences are known. Since all objective sentences are grounded, this sentence will not make any claims as to whether it is itself known or not. The idea is here that we have *protected* the occurrence of the subformula Kx by the formula $\mathit{Obj}\,(x)$ in such a way that quantification will effectively only be over the objective sentences. This strategy of protecting occurrences of Kx to avoid self-reference is the idea behind our definition of the protected sentences.

Definition 6.16. *A sentence φ in L is called* ***protected*** *if, for any variable x, the expression Kx only occurs in φ as part of subformulas of the form*

$$\forall x \, (\alpha(x) \to \beta(x)) \, ,$$

where $\alpha(x)$ represents a set of regular sentences. Otherwise φ is called ***unprotected****.*

Example 6.17 (The *About* predicate)**.** Consider again the sentence ψ from Example 6.10. This sentence is given by

$$\psi = \forall x \, (\mathit{About}\,(x, \mathit{striped}) \wedge \mathit{Obj}\,(x) \wedge \mathit{Lit}\,(x) \to \neg Kx) \, .$$

Define formulas $\alpha(x)$ and $\beta(x)$ by

$$\begin{aligned} \alpha(x) &= \mathit{About}\,(x, \mathit{striped}) \wedge \mathit{Obj}\,(x) \wedge \mathit{Lit}\,(x) \\ \beta(x) &= \neg Kx. \end{aligned}$$

Then ψ is the sentence $\forall x \, (\alpha(x) \to \beta(x))$. Since the subformula $\mathit{Obj}\,(x)$ represents the set of objective sentences of L, and since all objective sentences trivially are regular, $\alpha(x)$ must be representing a set of regular sentences. This proves that ψ is protected. If we succeed in proving that the reflection principles can consistently be instantiated with all protected sentences, then we have a proof that ψ can safely be reasoned about by our agents. We are going to prove this result in the following.

Example 6.18. Consider a *formal knower sentence* given by

$$\beta = \forall x_2 \, (D(\alpha, x_2) \to \neg K x_2) \, ,$$

where $\alpha(x_1) = \forall x_2 \, (D(x_1, x_2) \to \neg K x_2)$. A knower sentence on this form is guaranteed to exist by the diagonalisation lemma (Lemma 4.2). Since $D(x_1, x_2)$ represents the diagonalisation function, the formula $D(\alpha, x_2)$ must represent the singleton $\{\alpha(\alpha)\}$, which is equal to $\{\beta\}$. Since β is not regular, the formula $D(\alpha, x_2)$ represents a set of non-regular sentences. From this it follows that the sentence β is unprotected. Thus none of the self-referential knower sentences guaranteed to exist by the diagonalisation lemma are protected.

Example 6.17 above shows that there exists protected sentences which are not grounded. The converse is not true. If a sentence is grounded then it does not contain Kx as a subformula, so it trivially satisfies the condition for being protected. We have therefore proved the following.

Lemma 6.19. *The set of protected sentences of L is a proper extension of the set of grounded sentences of L.*

Since we already know the grounded sentences to be a proper extension of the regular ones (Lemma 5.27), we have the following chain of strict inclusions

$$\{\varphi \mid \varphi \text{ is regular}\} \subset \{\varphi \mid \varphi \text{ is grounded}\} \subset \{\varphi \mid \varphi \text{ is protected}\}.$$

We wish to bring attention to the fact that there is a certain *second-order flavour* to the protected sentences. We seem to have introduced two levels of sentences: the regular sentences and the protected sentences. The regular sentences are at the lower level, and these do not involve quantification over knowledge. The protected sentences are at the higher level, and they only involve quantification over knowledge concerning sentences at the lower level.

Definition 6.20. *To every formula φ in L we associate a natural number $q(\varphi)$, called its* ***non-regularity degree***. *The function q is defined recursively by*

(i) $q(Kx) = 1$, *for any variable x.*
(ii) $q(K\varphi) = \varphi$, *for any sentence φ in L.*
(iii) $q(\varphi) = 0$, *for any other atomic formula.*
(iv) $q(\neg\varphi) = q(\varphi)$.
(v) $q(\varphi \wedge \psi) = q(\varphi) + q(\psi)$.
(vi) $q(\forall x\varphi) = q(\varphi)$.

Example 6.21. The sentence $\forall x Kx$ has non-regularity degree one. By condition (ii) above, $K\forall xKx$ also has non-regularity degree one, even though Kx is not a subformula in $K\forall xKx$ (it appears only in coded form). The sentence $K\forall xKx \wedge \forall yKy$ has non-regularity degree 2, by condition (v) above. By condition (iii), $A\forall xKx$ has non-regularity degree zero if A is distinct from K. From conditions (ii)–(v) we see that any regular formula has non-regularity degree zero.

The following lemma shows that it is not possible to construct indirectly self-referential sentences in U_L^- without using unprotected sentences.

Lemma 6.22. *Let σ be an infinite path in G_L^-. Then σ contains infinitely many unprotected sentences.*

Proof. Assume to obtain a contradiction that there exists an infinite path σ in G_L^- containing only finitely many unprotected sentences. Then there exists a subpath σ' of σ containing only protected sentences.

Claim 1. Suppose $(\forall x\varphi(x), \varphi(\tau))$ is an edge on σ', where Kx occurs in $\varphi(x)$. Then the non-regularity degree of the end node $\varphi(\tau)$ is strictly less than the non-regularity degree of the start node $\forall x\varphi(x)$.

Proof of claim. If τ is not the Gödel code of a sentence in L, then it follows directly from Definition 6.20 that $\varphi(\tau)$ must have lower non-regularity degree than $\forall x \varphi(x)$. Assume therefore conversely that $\tau = \ulcorner \psi \urcorner$ for some sentence ψ. Then the edge is of the form $(\forall x \varphi(x), \varphi(\ulcorner \psi \urcorner))$. By choice of σ', the sentence $\forall x \varphi(x)$ is protected. It must therefore be on the form $\forall x\, (\alpha(x) \rightarrow \beta(x))$, where $\alpha(x)$ represents a set of regular sentences. Thus the edge is on the form

$$\left(\forall x\, (\alpha(x) \rightarrow \beta(x))\,, \alpha(\ulcorner \psi \urcorner) \rightarrow \beta(\ulcorner \psi \urcorner)\right).$$

By Lemma 6.15, we must have

$$Q \nvdash \alpha(\ulcorner \psi \urcorner) \rightarrow \beta(\ulcorner \psi \urcorner).$$

This implies

$$Q \nvdash \neg\alpha(\ulcorner \psi \urcorner). \tag{6.8}$$

Since $\alpha(x)$ is representing a set of natural numbers, the substitution instance $\alpha(\ulcorner \psi \urcorner)$ must be decidable in Q. From (6.8) we therefore get

$$Q \vdash \alpha(\ulcorner \psi \urcorner).$$

Since $\alpha(x)$ is representing a set of regular sentences, $\ulcorner \psi \urcorner$ must thus be the Gödel code of a regular sentence. The edge is therefore on the form

$$(\forall x \varphi(x), \varphi(\ulcorner \psi \urcorner))\,,$$

where ψ is regular. Since ψ is regular, it has non-regularity degree 0 (cf. Example 6.21). Thus the non-regularity degree of the end node $\varphi(\ulcorner \psi \urcorner)$ must be strictly less than the non-regularity degree of the start node $\forall x \varphi(x)$. This proves the claim.

Claim 2. The non-regularity degree is monotonically decreasing along σ'.

Proof of claim. Let (r_1, r_2) be any edge in σ'. We have to show that $q(r_2) \leq q(r_1)$. If (r_1, r_2) is a K-, $\neg$- or $\wedge$-edge, we immediately get $q(r_2) \leq q(r_1)$ by the clauses (ii), (iv) and (v), respectively, of Definition 6.20. The only case left to check is if (r_1, r_2) is a $\forall$-edge. So assume $r_1 = \forall x \varphi(x)$ and $r_2 = \varphi(\tau)$ for some formula φ and term τ. If $\varphi(x)$ does not contain Kx, then we obviously have $q(r_1) = q(r_2)$. Assume therefore that $\forall x \varphi(x)$ contains Kx. Then by Claim 1, $q(r_2) < q(r_1)$. Thus the claim is proved.

Claim 1 and Claim 2 now lead to a contradiction in the following way. Since by Claim 2, the non-regularity degree is monotonically decreasing along σ', there must be an infinite subpath σ'' of σ' on which the non-regularity degree is constant. By Lemma 5.28, the path σ'' contains a $\forall$-edge $(\forall x \varphi(x), \varphi(\tau))$, where Kx occurs in $\varphi(x)$. Now Claim 1 gives us that the end node of this edge has lower non-regularity degree than the start node, contradicting the choice of σ''. $\square$

Everything is now set for proving our main result.

Theorem 6.23 **(Second main result).** *Suppose S is the theory Q extended with a set of reflection principles instantiated over the protected sentences of L. Then S is consistent.*

Proof. Let M denote the set of protected sentences in L. Using Theorem 5.37, it suffices to prove that U_M has a model. Using Lemma 6.14, it furthermore suffices to prove that U_M^- has a model. This can be proved by showing that G_M^- is a non-paradoxical graph. We will show that G_M^- is well-founded. Then the required conclusion follows from Theorem 6.8. To prove that G_M^- is well-founded, assume the opposite. Then it must contain an infinite path $\sigma = (r_1, r_2, \dots)$. By Lemma 5.21, this path contains a K-edge (r_n, r_{n+1}). By choice of M, the end node r_{n+1} must be a protected sentence, since in G_M^- all K-edges with non-protected end nodes have been removed (Lemma 5.34). Let σ' be the subpath $(r_{n+1}, r_{n+2}, \dots)$ of σ. Every sentence in σ' must be protected, since the first node is protected and it is easy to see that all $\wedge$-, $\neg$-, $\forall$- and K-edges in G_M^- preserve protectedness (the K-edges do it simply because all K-edges with non-protected end nodes have been removed). Thus we immediately have a contradiction with Lemma 6.22, by which σ'' should contain infinitely many unprotected sentences. $\square$

6.2.2. Applying the Result. We will now through a number of examples show how the result above guarantees that agents can safely do a large amount of the introspective reasoning we are interested in.

Example 6.24 (The *About* predicate)**.** In Example 6.10 we considered various ways to formalise the sentence "the agent does not know anything about the striped block". We came to the conclusion that in order to obtain the intended interpretation, this sentence should be formalised as

$$\forall x \,(About(x, striped) \wedge Obj(x) \wedge Lit(x) \rightarrow \neg Kx)\,.$$

This sentence expresses that no objective literal concerning the striped block is known. In Example 6.17 we showed that the sentence is protected. By Theorem 6.23 it follows that we can consistently instantiate our reflection principles with the sentence, and thus agents will be allowed to reason about it consistently. Let us illustrate some of the consequences of this.

We can for instance define a formula $Ignorant\text{-}about(x_1, x_2)$ by

$$Ignorant\text{-}about(x_1, x_2) = \forall x_3 \,(About(x_2, x_3) \wedge Obj(x_3) \wedge Lit(x_3) \rightarrow \neg K_{x_1} x_3)\,.$$

This formula provides agents with an "ignorance about" modality. For any $n, m \in \mathbb{N}$, the sentence $Ignorant\text{-}about(n, m)$ expresses that the agent denoted by n is ignorant about the properties of the object denoted by m. The sentence $Ignorant\text{-}about(n, m)$ is a protected sentence, so reasoning about it can be done consistently. An agent can for instance express

"I do not have any knowledge about Malaysia"

by

$$K_0 \mathit{Ignorant\text{-}about}\,(0, \mathit{malaysia})\,.[2]$$

The agent will be allowed to reason about this sentence consistently, using any of the reflection principles A1–A7 (and T). An agent could also express the ignorance of other agents, as for instance in

$$K_{\mathit{bill}}\, \mathit{Ignorant\text{-}about}\,(\mathit{sue}, \mathit{cool\ jazz})\,,$$

expressing that Bill knows Sue to be ignorant about cool jazz. Conversely, to express that Bill knows Sue to know *something* about cool jazz, we could write

$$K_{\mathit{bill}}\,\neg \mathit{Ignorant\text{-}about}\,(\mathit{sue}, \mathit{cool\ jazz})\,.$$

The ability to express and reason about such sentences can for instance be advantageous when we want several agents to cooperate in carrying out various tasks. If an agent wishes to find another agent to help it solve a particular problem, then it can be important for this agent to be able to reason about which of the other agents have knowledge about the problem.

We now take a final look at the list of sentences considered in sections 4.3.1 and 6.1.2.

(i) $K\, \mathit{On}\,(\mathit{black}, \mathit{floor})$
(ii) $K\neg K\, \mathit{On}\,(\mathit{striped}, \mathit{white})$
(iii) $K\,(\neg K\, \mathit{On}\,(\mathit{striped}, \mathit{floor}) \wedge \neg K \neg \mathit{On}\,(\mathit{striped}, \mathit{floor}))$
(iv) $K\neg \exists x\,(\mathit{About}\,(x, \mathit{striped}) \wedge Kx)$
(v) $K_{\mathit{ole}}\, \forall x\,(\mathit{About}\,(x, \mathit{cool\ jazz}) \wedge K_{\mathit{sue}}\, x \rightarrow K_{\mathit{bill}}\, x)$
(vi) $K_{\mathit{john}}\, \exists x\,(K_{\mathit{bill}}\, x \wedge \neg K_{\mathit{john}}\, x)$
(vii) $\forall x \forall y\,(\mathit{IterateK}\,(\varphi, n, x) \wedge y < n \rightarrow K_y\,(x))$
(viii) $\forall x_1 \forall x_2\,(\mathit{Neg}\,(x_1, x_2) \rightarrow \neg\,(K x_1 \wedge K x_2))$

Of these sentences we know that only (i)–(iii) are regular and only (i)–(vi) are grounded. Let us take a look at the two remaining sentences, (vii) and (viii). We do this in the following two examples.

EXAMPLE 6.25 (Formalising common knowledge)**.** In Example 3.11 we showed that the sentence (vii) expresses that φ is common knowledge among the agents $0, \ldots, n-1$. Let us define $C(x_1)$ to be the following formula

$$C(x_1) = \forall x_2 \forall x_3\,(\mathit{IterateK}\,(x_1, n, x_2) \wedge x_3 < n \rightarrow K_{x_3}\,(x_2))\,.$$

[2]Recall that we use the constant symbol 0 as a default name for the agent itself.

Then for any sentence φ, the sentence $C(\varphi)$ expresses that φ is common knowledge (among agents $0, \dots, n-1$). The concept of common knowledge is no less problematic and no less vulnerable to problems of self-reference than the concept of knowledge itself. In fact, we can even construct a formal sentence ψ saying of itself that it is not common knowledge. This follows from the diagonalisation lemma (Lemma 4.2), which ensures the existence of a sentence ψ satisfying

$$Q \vdash \psi \leftrightarrow \neg C(\psi).$$

This sentence gives rise to the same problems as a formal knower sentence. We can prove a contradiction from the sentence in the same way as we proved a contradiction from the formal knower sentence in Montague's theorem (Theorem 4.5). In other words, it is not always safe to reason about common knowledge.

Let us take a closer look at the formula $C(x_1)$. It can be rewritten as

$$\forall x_3 \forall x_2 \left(\mathit{IterateK}\,(x_1, n, x_2) \rightarrow (x_3 < n \rightarrow K_{x_3}\,(x_2)) \right).$$

Let φ be any sentence in L. Then $C(\varphi)$ is the sentence

$$\forall x_3 \forall x_2 \left(\mathit{IterateK}\,(\varphi, n, x_2) \rightarrow (x_3 < n \rightarrow K_{x_3}\,(x_2)) \right).$$

From this we see that $C(\varphi)$ is protected if and only if $\mathit{IterateK}\,(\varphi, n, x_2)$ represents a set of regular sentences. Let us therefore try to see which conditions we have to put on φ in order for $\mathit{IterateK}\,(\varphi, n, x_2)$ to represent such a set. The formula $\mathit{IterateK}\,(x_1, x_2, x_3)$ is defined to represent the relation given by

$$\begin{aligned} \mathsf{IterateK} = \{ (\ulcorner \varphi \urcorner, n, \ulcorner K_{i_1} \cdots K_{i_m} \varphi \urcorner) \mid \varphi \text{ is a sentence, } m \geq 0, \\ \text{and } i_1, i_2, \dots, i_m < n \}. \end{aligned}$$

Since $K_{i_1} \cdots K_{i_m} \varphi$ is regular if and only if φ is regular, $\mathit{IterateK}\,(\varphi, n, x_2)$ will be representing a set of regular sentences if and only φ is regular. The sentence $C(\varphi)$ is therefore protected if and only if φ is regular!

It now follows from Theorem 6.23 that our agents can consistently reason about the common knowledge of regular sentences. This is probably sufficient for most practical purposes. It allows agents to express common knowledge of facts such as "the Emperor is naked", "we will attack at dawn" [Fagin *et al.*, 1995], "at least one child has mud on his forehead" [Fagin *et al.*, 1995], etc. That Theorem 6.23 only guarantees that the agents can consistently reason about common knowledge of *regular sentences* does not seem to be a problem for practical purposes, since we have anyway no interest in allowing them to express sentences such as "this sentence is not common knowledge".

The main advantage of our approach to common knowledge is that common knowledge is expressed exclusively in terms of the K predicate, whereas the traditional operator approach requires extra symbols and axiom schemes.

Even if we do not mind about introducing new axioms for common knowledge, our approach still has the advantage that most other modalities can be reduced to the K predicate as well. This includes modalities such as "knowing about" and "ignorance about" considered above. To express all of these modalities, we only need the language of arithmetic extended with the single predicate symbol K.

Example 6.26. In this example we consider the sentence φ given by

$$\varphi = \forall x_1 \forall x_2 \left(\mathit{Neg}\left(x_1, x_2\right) \rightarrow \neg \left(K x_1 \wedge K x_2\right)\right)$$

This sentence is (viii) above. We now take K to denote *belief* rather than *knowledge*, so the sentence expresses that there is no sentence φ such that both φ and $\neg\varphi$ are believed. In other words, it expresses that the agent in question has no contradictory beliefs. The sentence is unprotected. Thus our consistency result, Theorem 6.23, does not guarantee that the sentence can be reasoned about consistently. Suppose U is a first-order agent theory constituting the knowledge base of the agent in question. We assume that U contains a set of reflection principles, but that these—to ensure consistency—are only instantiated over the protected sentences. In this case the agent will not be able to make the inference from for instance $K\varphi$ to

$$K \neg \left(K\, \mathit{On}\left(\mathit{striped}, \mathit{floor}\right) \wedge K \neg \mathit{On}\left(\mathit{striped}, \mathit{floor}\right)\right),$$

since φ is not protected.

Let $\mathit{Reg}\left(x_1\right)$ be a formula representing the set of regular sentences in L. Consider the sentence ψ given by

$$\psi = \forall x_1 \forall x_2 \left(\mathit{Neg}\left(x_1, x_2\right) \wedge \mathit{Reg}\left(x_1\right) \wedge \mathit{Reg}\left(x_2\right) \rightarrow \neg \left(K x_1 \wedge K x_2\right)\right).$$

This sentence expresses that the agent has no contradictory *regular* beliefs. We immediately see that ψ is a protected sentence. So the agent will be able to make the inference from $K\psi$ to

$$K \neg \left(K\, \mathit{On}\left(\mathit{striped}, \mathit{floor}\right) \wedge K \neg K\, \mathit{On}\left(\mathit{striped}, \mathit{floor}\right)\right)$$

by a protected instance of reflection principle A3.

The example shows that even if the agent is prohibited from reasoning about general statements such as "I have no contradictory beliefs", it can still reason about restricted statements such as "I have no contradictory *regular* beliefs". In many cases this will be sufficient. It will for instance allow an agent to express and reason about sentences such as "I do not have any contradictory beliefs about Malaysia", where we use the formalisation of "about" given above.

6.3. Chapter Notes

Most of the material in this chapter up to and including Theorem 6.8 is closely related to similar developments in connection with logic programming semantics [Fitting, 2002a] and formal theories of truth [Visser, 1989]. Our presentation of these parts is adapted from Bolander [2003c]. The other parts of the chapter are original. A preliminary version of Theorem 6.9 appears in Bolander [2002b].

Our results concerning consistency of agent theories do not seem to correspond to any results developed within logic programming. There is, however, a very close correspondence in the underlying ideas: To obtain results on consistency, we represent patterns of semantic dependency by graphs and try to see which properties of these graphs are sufficient to guarantee consistency. Results of this type within logic programming can for instance be found in Apt *et al.* [1988], Cortesi and Filè [1993], Kunen [1987; 1989] and Sato [1990]. The main difference between our work and the work carried out in logic programming is that in logic programming the focus is on giving a suitable semantic treatment of *negation* that will ensure consistency, whereas in our work the focus has been on giving a suitable semantic treatment of *quantification*. Roughly, the difference is between avoiding paradoxes by making self-reference innocuous (by restricting negation) and by avoiding self-reference altogether (by restricting quantification).

Both of our consistency results are based on proving certain graphs to be well-founded. From this fact one might suspect that the results can be proved more directly by transfinite recursion. It is however not entirely obvious how such a thing should be done.

We have not been discussing the possibility of strengthening Theorem 6.23. It seems obvious to try to prove the consistency of even larger sets of instances of the reflection principles. It can however be shown that if we replace "a set of regular sentences" by "a set of grounded sentences" in the definition of the protected sentences, then Theorem 6.23 will no longer hold. The argument is fairly simple, but involves some work in giving a slightly more general version of the diagonalisation lemma.

Using Zorn's lemma and the compactness theorem for first-order logic, it can be proved that there must exist a *maximal* set of instances of any combination of the reflection principles. However, McGee [1992] has shown that no such maximal set can be recursively axiomatisable. It is therefore not sensible to look for maximal sets of consistent instances, but rather one should look for any consistent set of instances that will be sufficient for expressing the things one wishes to express.

Conclusion

In the thesis we have been studying the possibility of allowing artificial intelligence agents to do strong introspective reasoning, that is, to reason about their own knowledge and belief in non-trivial ways. This subject belongs to the research area of knowledge representation and reasoning. One of the main problems within the subject is to find ways to ensure that introspective agents are not allowed to perform self-contradictory reasoning. This problem has been the focus of the present work.

The thesis makes two major research contributions. The first is to introduce graphs and graph-based methods into the area of syntactical treatments of knowledge. The second is to apply these graph-based methods to strengthen the previously known results on the consistency of such syntactical treatments. Graphs and graph-based methods related to ours have been studied in the areas of logic programming semantics within computational logic and formal theories of truth within philosophical logic. However, it is the first time that such graphs have been defined for the full language of first-order predicate logic and it is the first time that such methods have been applied in solving inconsistency problems for syntactical treatments of knowledge.

What we have done is to develop a general framework for representing patterns of semantic dependency by graphs. Through such graphs we have been able to give a precise characterisation of *self-reference*. This has allowed us to study self-referential phenomena occuring in the representation and reasoning of introspective agents. Through these studies, we have been able to propose new ways to prevent agents from performing self-contradictory reasoning. This has led us to our main result, Theorem 6.23. The theorem shows that agents can safely formulate and reason about introspective pieces of knowledge such as: “Everything Bill knows about the game, I know”, “I believe that none of my (regular) beliefs are contradictory” and “I do not know anything about Malaysia”.

We began the thesis by giving an informal introduction to the research area and research problem of our work. This was done in Chapter 1. In Chapter 2 we argued that the traditional modal logic approach to formalising the knowledge of agents is insufficient to properly deal with introspective knowledge.

This led us to consider first-order predicate logic as an alternative framework. In Chapter 3 we presented the details of this alternative framework. Then, in Chapter 4, we showed that the framework suffers from inconsistency problems related to the possibility of reasoning about self-referential sentences. In Chapter 5 we developed our graph-based method and in Chapter 6 we demonstrated how it could be applied to circumvent these inconsistency problems. This finally led us to propose a restricted framework for representing and reasoning about introspective knowledge. The framework effectively avoids the inconsistency problems, but still retains the expressive power needed for most kinds of introspective reasoning.

Bibliography

[Aczel and Feferman, 1980] Peter Aczel and Solomon Feferman. Consistency of the unrestricted abstraction principle using an intensional equivalence operator. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 67–98. Academic Press, London, 1980.

[Aczel, 1988] Peter Aczel. *Non-Well-Founded Sets.* Stanford University Center for the Study of Language and Information, 1988.

[Andréka *et al.*, 1995] Hajnal Andréka, Johan van Benthem, and Istvan Németi. Back and forth between modal logic and classical logic. *Journal of the Interest Group in Pure and Applied Logics*, 3(5):685–720, 1995.

[Apt and Bol, 1994] Krzysztof R. Apt and Roland N. Bol. Logic programming and negation: a survey. *Journal of Logic Programming*, 19(20):9–71, 1994.

[Apt *et al.*, 1988] K. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 89–142, Los Altos, CA, 1988. Morgan Kaufmann.

[Asher and Kamp, 1986] Nicholas M. Asher and Johan A. W. Kamp. The knower's paradox and representational theories of attitudes. In *Theoretical aspects of reasoning about knowledge (Monterey, Calif., 1986)*, pages 131–147. Morgan Kaufmann, 1986.

[Attardi and Simi, 1995] Guiseppe Attardi and Maria Simi. A formalization of viewpoints. *Fundamenta Informaticae*, 23(2-4):149–173, 1995.

[Baral and Gelfond, 1994] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19(20):73–148, 1994.

[Bartlett, 1992] Steven J. Bartlett, editor. *Reflexivity—A Source-Book in Self-Reference.* North-Holland, Amsterdam, 1992.

[Barwise and Etchemendy, 1987] Jon Barwise and John Etchemendy. *The Liar—An Essay on Truth and Circularity.* Oxford University Press, 1987.

[Barwise and Moss, 1996] Jon Barwise and Lawrence Moss. *Vicious circles.* CSLI Publications, 1996.

[Beck, 2002] Andreas Beck. *The Liar Lies and Snow is White—A consistent theory of truth for semantically closed formal languages.* PhD thesis, München, 2002.

[Bolander, 2002a] Thomas Bolander. Maximal introspection of agents. *Electronic Notes in Theoretical Computer Science*, 70(5), 2002. Elsevier Science. 16 pages.

[Bolander, 2002b] Thomas Bolander. Restricted truth predicates in first-order logic. In *The LOGICA Yearbook 2002*, pages 41–55. Filosofia, Prague, 2002.

[Bolander, 2002c] Thomas Bolander. Self-reference and logic. *Phi News*, 1:9–44, 2002. PhiLog, Kluwer Academic Publishers.

[Bolander, 2003a] Thomas Bolander. From logic programming semantics to the consistency of syntactical treatments of knowledge and belief. In *Proceedings of IJCAI-03 (Eighteenth*

International Joint Conference on Artificial Intelligence), pages 443–448. Morgan Kaufmann, Elsevier Science, 2003.

[Bolander, 2003b] Thomas Bolander. Which patterns of semantic dependency are paradoxical?, I. 2003. 20 pages. Submitted for publication.

[Bolander, 2003c] Thomas Bolander. Which patterns of semantic dependency are paradoxical?, II. 2003. 22 pages. Submitted for publication.

[Cantor, 1891] Georg Cantor. Über eine elementare Frage der Mannigfaltigkeitslehre. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 1:75–78, 1891. Reprinted in [Cantor, 1932].

[Cantor, 1932] Georg Cantor. *Gesammelte Abhandlungen*. Springer Verlag, 1932.

[Carlucci Aiello *et al.*, 1995] Luigia Carlucci Aiello, Marta Cialdea, Daniele Nardi, and Marco Schaerf. Modal and meta languages: consistency and expressiveness. In *Metalogics and logic programming*, pages 243–265. MIT Press, 1995.

[Cook, 2002] Roy T. Cook. Parity and paradox. In *The LOGICA Yearbook 2002*, pages 69–83. Filosofia, Prague, 2002.

[Cortesi and Filé, 1993] Agostino Cortesi and Gilberto Filé. Graph properties for normal logic programs. *Theoretical Computer Science*, 107(2):277–303, 1993.

[Davies, 1990] Nick Davies. A first order logic of truth, knowledge and belief. *Lecture Notes in Artificial Intelligence*, 478:170–179, 1990.

[des Rivières and Levesque, 1988] Jim des Rivières and Hector J. Levesque. The consistency of syntactical treatments of knowledge. *Computational Intelligence*, 4:31–41, 1988.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[Fasli, 2003] Maria Fasli. Reasoning about knowledge and belief: A syntactical treatment. *Logic Journal of the IGPL*, 11(2):245–282, 2003.

[Feferman, 1962] Solomon Feferman. Transfinite recursive progressions of axiomatic theories. *The Journal of Symbolic Logic*, 27:259–316, 1962.

[Feferman, 1984] Solomon Feferman. Toward useful type-free theories I. *The Journal of Symbolic Logic*, 49(1):75–111, 1984.

[Feferman, 1991] Solomon Feferman. Reflecting on incompleteness. *The Journal of Symbolic Logic*, 56(1):1–49, 1991.

[Fitting and Mendelsohn, 1998] Melvin Fitting and Richard L. Mendelsohn. *First-order modal logic*, volume 277 of *Synthese Library*. Kluwer Academic Publishers Group, Dordrecht, 1998.

[Fitting, 2002a] Melvin Fitting. Fixpoint semantics for logic programming—a survey. *Theoretical Computer Science*, 278(1-2):25–51, 2002.

[Fitting, 2002b] Melvin Fitting. *Types, Tableaus, and Gödel's God*. Kluwer Academic Publishers, 2002.

[Friedman and Sheard, 1987] Harvey Friedman and Michael Sheard. An axiomatic approach to self-referential truth. *Annals of Pure and Applied Logic*, 33(1):1–21, 1987.

[Gaifman, 1992] Haim Gaifman. Pointers to truth. *Journal of Philosophy*, 89(5):223–261, 1992.

[Gallin, 1975] Daniel Gallin. *Intensional and higher-order modal logic—With applications to Montague semantics*. North-Holland Publishing Co., Amsterdam, 1975.

[Gödel, 1931] Kurt Gödel. Über formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931. Reprinted in [Gödel, 1986].

[Gödel, 1986] Kurt Gödel. *Collected works. Vol. I.* Oxford University Press, 1986. Publications 1929–1936, Edited and with a preface by Solomon Feferman.

[Grant *et al.*, 2000] John Grant, Sarit Kraus, and Donald Perlis. A logic for characterizing multiple bounded agents. *Autonomous Agents and Multi-Agent Systems*, 3(4):351–387, 2000.

[Grim, 1993] Patrick Grim. Operators in the paradox of the knower. *Synthese*, 94(3):409–428, 1993.

[Gupta and Belnap, 1993] Anil Gupta and Nuel Belnap. *The Revision Theory of Truth.* MIT Press, 1993.

[Harnish, 1993] Robert N. Harnish, editor. *Basic Topics in the Philosophy of Language.* Prentice-Hall, 1993.

[Hintikka, 1962] Jaakko Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Cornell University Press, 1962.

[Hughes and Cresswell, 1996] G. E. Hughes and M. J. Cresswell. *A new introduction to modal logic.* Routledge, London, 1996.

[Kaplan and Montague, 1960] David Kaplan and Richard Montague. A paradox regained. *Notre Dame Journal of Formal Logic*, 1(3):79–90, 1960.

[Kerber, 1998] Manfred Kerber. On knowledge, strings, and paradoxes. *Lecture Notes in Artificial Intelligence*, 1489:342–354, 1998.

[Kleene, 1964] S. C. Kleene. *Introduction to Metamathematics.* North-Holland, 1964.

[Konolige, 1982] Kurt Konolige. A first-order formalization of knowledge and action for a multiagent planning system. *Machine Intelligence*, 10:41–72, 1982.

[Konolige, 1988] Kurt Konolige. Reasoning by introspection. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection.* North-Holland, 1988.

[Kraus *et al.*, 1991] Sarit Kraus, Donald Perlis, and John Horty. Reasoning about ignorance: A note on the Bush-Gorbachov problem. *Fundamenta Informaticae*, 15(3–4):325–332, 1991.

[Kripke, 1963] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Kripke, 1975] Saul Kripke. Outline of a theory of truth. *The Journal of Philosophy*, 72:690–716, 1975. Reprinted in [Martin, 1984].

[Kunen, 1987] Kenneth Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.

[Kunen, 1989] Kenneth Kunen. Signed data dependencies in logic programs. *Journal of Logic Programming*, 7(3):231–245, 1989.

[Lakemeyer, 1992] Gerhard Lakemeyer. On perfect introspection with quantifying-in. In *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the Fourth Conference (TARK 1992)*, pages 199–213. Morgan Kaufmann, 1992.

[Levesque and Lakemeyer, 2000] Hector J. Levesque and Gerhard Lakemeyer. *The Logic of Knowledge Bases.* MIT Press, 2000.

[Levesque, 1984] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984. Reprinted in [Bartlett, 1992].

[Lloyd, 1987] J. W. Lloyd. *Foundations of logic programming.* Springer-Verlag, Berlin, second edition, 1987.

[Martin, 1984] Robert L. Martin, editor. *Recent Essays on the Liar Paradox.* Oxford University Press, 1984.

[McCarthy, 1979] John McCarthy. Ascribing mental qualities to machines. In *Philosophical Perspectives in Artificial Intelligence*, pages 161–195. Humanities Press, 1979.

[McCarthy, 1996] John McCarthy. Making robots conscious of their mental states. In *Machine Intelligence 15*, pages 3–17. Oxford University Press, 1996.

[McCarthy, 1997] John McCarthy. Modality si! Modal logic, no! *Studia Logica*, 59(1):29–32, 1997.

[McGee, 1985] Vann McGee. How truthlike can a predicate be? A negative result. *Journal of Philosophical Logic*, 14(4):399–410, 1985.

[McGee, 1992] Vann McGee. Maximal consistent sets of instances of Tarski's schema (T). *Journal of Philosophical Logic*, 21(3):235–241, 1992.

[Mendelson, 1997] Elliott Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 4 edition, 1997.

[Montague, 1963] Richard Montague. Syntactical treatments of modality, with corollaries on reflection principles and finite axiomatizability. *Acta Philosophica Fennica*, 16:153–166, 1963.

[Moreno, 1998] Antonio Moreno. Avoiding logical omniscience and perfect reasoning: a survey. *AI Communications*, 11(2):101–122, 1998.

[Morreau and Kraus, 1998] Michael Morreau and Sarit Kraus. Syntactical treatments of propositional attitudes. *Artificial Intelligence*, 106(1):161–177, 1998.

[Perlis and Subrahmanian, 1994] Donald Perlis and V. S. Subrahmanian. Meta-languages, reflection principles and self-reference. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 323–358. Oxford University Press, 1994.

[Perlis, 1985] Donald Perlis. Languages with self-reference I. *Artificial Intelligence*, 25:301–322, 1985.

[Perlis, 1988] Donald Perlis. Languages with self-reference II. *Artificial Intelligence*, 34:179–212, 1988.

[Priest, 1989] Graham Priest. Reasoning about truth. *Artificial Intelligence*, 39(2):231–244, 1989.

[Priest, 1991] Graham Priest. Intensional paradoxes. *Notre Dame Journal of Formal Logic*, 32(2):193–211, 1991.

[Sato, 1990] Taisuke Sato. Completed logic programs and their consistency. *Journal of Logic Programming*, 9(1):33–44, 1990.

[Smullyan, 1984] Raymond M. Smullyan. Chameleonic languages. *Synthese*, 60(2):201–224, 1984.

[Tarski, 1944] Alfred Tarski. The semantic conception of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, 4:341–376, 1944. Reprinted in [Harnish, 1993].

[Tarski, 1956] Alfred Tarski. The concept of truth in formalized languages. In *Logic, semantics, metamathematics—Papers from 1932 to 1938*. Hackett Publishing Co., 1956.

[Thomason, 1980] Richmond H. Thomason. A note on syntactical treatments of modality. *Synthese*, 44(3):391–395, 1980.

[Turner, 1990] Raymond Turner. *Truth and Modality for Knowledge Representation*. Pitman Publishing Ltd., 1990.

[Visser, 1989] Albert Visser. Semantics and the liar paradox. In *Handbook of philosophical logic*, volume 4, pages 617–706. D. Reidel Publishing Company, 1989.

[Whitsey, 2003] Mark Whitsey. Logical omniscience: A survey. Unpublished paper, 2003.

[Yablo, 1982] Steve Yablo. Grounding, dependence, and paradox. *Journal of Philosophical Logic*, 11(1):117–137, 1982.

[Yablo, 1985] Stephen Yablo. Truth and reflection. *Journal of Philosophical Logic*, 14(3):297–349, 1985.

[Yablo, 1993] Stephen Yablo. Paradox without self-reference. *Analysis*, 53(4):251–252, 1993.

Index