Feasible Strategies in Alternating-time Temporal Epistemic Logic

Geert M. Jonker

Doctoraalscriptie Informatica
Begeleiders: Prof. Dr. W. van der Hoek en Prof. Dr. J.-J. Ch. Meyer
Faculteit Wiskunde en Informatica
Universiteit Utrecht
Examendatum: 26 september 2003

Summary

Alternating-time Temporal Logic (ATL) is a temporal logic that allows for reasoning about strategies that agents may follow to guarantee the satisfaction of certain conditions in the future. Alternating-time Temporal Epistemic Logic (ATEL) is an extension of ATL, allowing for expression of epistemic conditions as well. We propose an adaptation of ATEL called Feasible ATEL, where a natural consequence of epistemicality is reflected in the definition of strategies. We claim that in many real-life scenario's, Feasible ATEL is more suitable for appropriately describing the state of affairs than ATEL. We go on to study several knowledge related scenarios and introduce operators capturing a number of essential epistemic principles related to strategies.

Acknowledgments

I would like to thank Wiebe van der Hoek for inviting me to come to Liverpool. I'm not the bravest of people, but with Wiebe as my supervisor I knew I was going to be okay. I've always considered him not only my supervisor, but also a friend.

I also thank Mike Wooldridge for providing for me the opportunity to study at his department, as well as for support and guidance he gave me.

Special thanks to Tabasum for her extraordinary deed of encouraging me to go to Liverpool whilst being my girlfriend, as well as for the love and support she gave me during the time there.

Last but not least I thank Sieuwert for being a fine intellectual sparring partner and, moreover, a good friend. He introduced me to a lot of people, the unexpected pleasure of karate and several useful insights into the world of academic life. Also his critical but usually just remarks on my academic endeavours were greatly appreciated.

Preface

It was in 1998 that I received my first course in logic. Although the teacher was making a very big effort to present it as boring as possible, it did attract my attention. There was something about logic that made it very appealing, charming almost. Maths had always been one of my better subjects, but the sheer juggling with numbers never had the same effect. Instead I would end up in a philosophical discussion with my maths teacher about the difference between $\frac{1}{2}$ and 0.5[1]. Philosophic and fundamental issues always interested me... and logic proves to be full of them.

Logic can be defined as the study of *correct reasoning*. It is in a way very similar to human reasoning since it takes principles from human reasoning and captures them in its own language of letters and symbols. It is at the same time very different from human reasoning since it becomes a reasoning system by itself which can draw very different conclusions than we human beings would do. In the process of designing a logic, our own way of reasoning is often the source from which the needed rules and principles are extracted. At the other hand, a logical system can reveal principles that make us think about ourselves, about whether we would reason in that way or not.

Suppose we realize that, if apes are mammals, it is a correct way of reasoning to conclude that an animal that is not a mammal is certainly not an ape. This principle of reasoning can be expressed as $(p \to q) \to (\neg q \to \neg p)$. If we take p to denote the fact that "animal x is an ape" and q to denote the fact that "animal x is a mammal" and $p \to q$ to be read as "if p then q", we arrive at our initial assertion again. The logical principle can now also be used for other situations. Suppose we also realize the fact that

[1] A bitter dispute which remains unresolved until today.

if it rains, we become wet. Using the logical principle above, we can now derive the fact that if we don't become wet, it doesn't rain. Unfortunately not a very plausible conclusion, since we could stay dry during the rain if we have an umbrella. Something went wrong in our attempt to reason correctly. We could say that the principle we used is incorrect. Apparently there are situations where the principle does not apply. Apparently this is not the way humans reason. We could also reconsider the fact "if it rains we become wet". If we make an assertion like this, do we really mean it as we say it? Or do we actually mean "if it rains and we're not holding an umbrella, we become wet"? In that case, the principle still holds. We could still consider it to be a form of "correct reasoning".

Often the line between correct and incorrect reasoning is not so easy to define. In *classical logic*, a statement p is always either true or false. The formula $p \vee \neg p$ always holds. Furthermore, we can prove the truth of p by proving that its negation, $\neg p$ cannot be true. *Intuitionistic logic* rejects this form of reasoning. Its claim is that humans do not use this kind of reasoning and therefore it can not be called "correct". The set of formulas provable in intuitionistic logic is therefore different than that of classical logic. In fact, there exist a great number of logics which are different from each other because of the inclusion or rejection of certain principles of reasoning. Usually, the reasons for existence are not so much different views of the term "correct reasoning", but rather the different aspects of reasoning which can be expressed and examined with a particular logic. Much depends on the requirements given by the context in which the logic is used.

In the fields of computer science and artificial intelligence, logics have always played an important role. The requirements in this area have led to the use and further developing of several families of logics, including temporal logic in which *time* is formalized and epistemic logic, in which *knowledge* is formalized. Particularly the latter has always drawn my attention, since reasoning about your own as well as others' knowledge is a fascinating area giving rise to interesting scenarios with captivating names as *Muddy Children* and *Coordinated Attack*.

This thesis is concerned with a logic which is both temporal and epistemic and in addition uses the concept of *strategies*. This logic, ATEL, is aimed at *multi agent systems*, a new programming paradigm that is emerg-

ing at the moment. The emphasis of this thesis lies on a flaw in this logic and its proposed solution. Furthermore we explore the effects of some different aspects of knowledge and look into what a "winning position" might be. We also touch on the subject of *program synthesis*, a principle that was mentioned in the paper introducing ATEL.

This thesis is organized as follows: After the introduction in the first chapter, we introduce in the second chapter the logics that are fundamental to our research: modal, temporal and epistemic logic. We also introduce the concept of *distributed systems*. In the third chapter we introduce ATL, a temporal logic which uses the concept of strategies, as well as ATEL, which enriches ATL with epistemic operators. We also give an example of a game with incomplete knowledge to find a flaw in the definition of strategies in ATEL, leading to counter-intuitive scenario's. In the fourth chapter, we propose a different version of ATEL to correct the flaw: Feasible ATEL. We show how this logic leads to more intuitive results. We furthermore discuss the concepts of *perfect recall* and *synchrony*, and give an idea of how models for feasible ATEL can be constructed with little effort for these mental capabilities. We then explore the nature of feasible ATEL further by examining some other thorny scenarios. We identify the need for and introduce new operators for knowledge of strategies. Finally we look into the concept of a winning position.

Contents

Summary **i**

Acknowledgments **iii**

Preface **v**

1 Introduction **1**

2 Background **5**
2.1 Modal Logic 5
2.2 Temporal Logic 8
2.3 Epistemic Logic 10
2.4 Knowledge in Distributed Systems 12

3 Alternating-time Logics **15**
3.1 ATL 15
3.1.1 Concurrent Game Structures 16
3.1.2 Syntax and Semantics 18
3.2 ATEL 19
3.2.1 Syntax and Semantics 20
3.2.2 Applications of ATEL 21
3.3 The game "King Queen Ace" 22
3.4 Some properties reviewed 25

4 Feasible ATEL **27**
4.1 Feasible ATEL 27
4.1.1 Concurrent Epistemic Game Structures 28

4.1.2 Syntax en Semantics 29
4.2 The game “King Queen Ace” again 31
4.3 Some properties reviewed again 32
4.4 A problematic case 34
4.5 Feasible ATEL and Perfect Recall 38
4.6 Knowing the strategy 43
4.7 Knowledge of strategies in groups 46
4.8 Winning position 55

Conclusion **59**

List of figures **63**

List of lemmas **65**

Bibliography **66**

Chapter 1

Introduction

In the past few decades, computers have become increasingly important in the world around us. We use them to write our letters, to read the news, to buy our clothes. They are in the elevator, in our cd-player and in our telephones. More and more tasks are taken out of our hands and being performed by them.

As technology advances, the capabilities of computers grow. And as their capabilities grow, larger and larger tasks are given to them. The administration of a company, a telephone network, the stock exchange. The growing size of software applications has had a number of effects. Single computers don't have the capacity anymore to host large applications. The many lines of code increase the frequency of errors and crashes. Also, the data and users that an application needs to involve are spread over a large area. All these reasons have led to the development of *distributed systems* — applications that consist of several processes executing independently and working together. Distributed systems utilize the combined capacity of several computers, their processes can be restarted in case of crashes and data and users can be reached at any location.

Some tasks performed by computers are more vital than others. While a crashing word processor is merely an annoyance to us, a crashing program in a hospital can mean the difference between life and death. As the impact of computer programs increases, the question of whether a program does its task correctly becomes more important. Often the reliability of a program is determined by testing. In order to test a program, one needs a description

of the properties we want the program to have. With small programs, these descriptions can be specified intuitively. With large or distributed programs however, giving a clear specification of the properties we want our programs to have can be very difficult. Often natural language is not suitable for this task. In that case we need to use logics to be able to specify the properties we want to test.

Testing can have a number of disadvantages. It can be costly, it can be too big a task to perform, or the degree of certainty required can not be met by testing. In those cases the *correctness* of a program or the satisfaction of certain essential properties need to be *proven*. In order to do this one again needs appropriate logics to construct the arguments that prove the programs correctness. This is referred to as *program verification*.

Multi-agent systems is a new technology that is currently emerging in the area of distributed software. An *intelligent agent* is a computer program equipped with a certain level of artificial intelligence. They are designed to be able to sense and observe information, reason and make decisions for themselves, and communicate and work together with other agents. Multi-agent systems are expected to be able to handle complex and mission critical tasks such as air traffic control, as well as comparatively small task such as email filtering.

Verification of multi-agent systems properties is receiving increasing attention [AHK97, vdHW02]. For this purpose several logics have been developed formalizing different principles related to intelligent agents. These logics involve principles from the field of program correctness, such as *time* and *action*. As multi-agent systems are dynamic, they change over time. The family of logics concerning time is referred to as *temporal logic*. The evolution of a system is caused by the behaviour of the agents, which is visible by the actions they perform. Actions are formalized in *dynamic logic*. Furthermore, the concept of knowledge or belief is essential to a logic for multi-agent systems, since it is one of the basic notions of agent theory [WJ95]. The family of logics concerning knowledge or belief is referred to as *epistemic logic*.

In this paper we will examine a logic that involves time, actions and knowledge, and in addition the concept of *strategies*. In many cases, for instance security related situations, one is interested in whether an agent or

a group of agent can guarantee to achieve a certain goal. Whether this is the case can be viewed as the question of whether they have a strategy to attain their goal.

For a thorough understanding of our subject we will in the next chapter introduce the logics fundamental to our research: temporal logic and epistemic logic. These two logics however have been derived from another logic called *modal logic*, which we will therefore introduce first.

Chapter 2

Background

2.1 Modal Logic

Modal Logic was invented by Lewis in 1918 [Lew18]. He sought to create a logic that didn't suffer from two 'paradoxes' of classical propositional logic: "A false proposition implies any proposition" and "A true proposition is implied by any proposition". He therefore made a distinction between *necessary* propositions, *impossible* propositions and *contingent* propositions. A necessary proposition is a proposition that is bound to be true, or necessarily true. Similarly an impossible proposition is one that is bound to be false. A contingent proposition is one that can or happens to be either true or false. He then defined the modal operator $\Diamond$ for possibility, in the way that $\Diamond p$ is read as "it is possible that p is true" or "p is not impossible". With this operator he was able to define 'strict implication' which was very similar to classical implication but without the paradoxes described above.

The ideas of necessity and possibility have been adopted by many scientists since. One of them introduced the modal operator $\Box$ for necessity [Bar46]; $\Box p$ is read as "p is necessarily true". The two operators relate as follows: $\Box\varphi \;\hat{=}\; \neg\Diamond\neg\varphi$.

Although Lewis was mainly concerned with the deductive behaviour of his modal operators, the term 'modal logic' is now also used more broadly to cover a family of logics with similar rules. A list describing the best known of these logics follows.

Logic	**Symbols**	**Expressions symbolized**
Modal Logic	$\Box$	It is necessary that ...
	$\Diamond$	It is possible that ...
Intuitionistic Logic	$\Box$	It is provable that ...
Deontic Logic	O	It is obligatory that ...
	P	It is permitted that ...
	F	It is forbidden that ...
Temporal	G, $\Box$	It will always be the case that ...
	F, $\Diamond$	It will eventually be the case that ...
	N, $\bigcirc$	At the next moment, it will be the case that ...
	H	It has always been the case that ...
	P	It was the case that ...
Epistemic Logic	K_a, $\Box_a$	a knows that ...
	M_a, $\Diamond_a$	a considers possible that ...
Doxastic Logic	B_a	a believes that ...

The most elementary logic of the modal family is a weak logic called K (after Saul Kripke). Let Φ be the set of atomic propositions $p_0, p_1, p_2, ...$ The language $\mathcal{ML}$ of K is defined by:

1. If $p \in \Phi$ then $p \in \mathcal{ML}$.

2. If $\varphi, \psi \in \mathcal{ML}$ then $\neg\varphi, (\varphi \wedge \psi), \Box\varphi \in \mathcal{ML}$.

To this, the following abbreviations are added:

$$\begin{aligned} \bot &\hat{=} p_0 \wedge \neg p_0 \\ \varphi \vee \psi &\hat{=} \neg(\neg\varphi \wedge \neg\psi) \\ \varphi \rightarrow \psi &\hat{=} \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi &\hat{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \\ \Diamond\varphi &\hat{=} \neg\Box\neg\varphi \end{aligned}$$

The calculus of K consists of the axioms of propositional logic and the inference rules of *Substitution* and *Modus Pones*, extended by the Distribution Axiom:

(AK) $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

and the inference rule of Necessitation:

(RN) given φ, derive $\Box\varphi$

According to RN, any theorem of K is necessary.

Every other modal logics can be obtained by extending this system with other axioms. Well known axioms include:

(AM) $\Box\varphi \to \varphi$
(A4) $\Box\varphi \to \Box\Box\varphi$
(A5) $\Diamond\varphi \to \Box\Diamond\varphi$

by which the following logics can be defined:

$$\begin{aligned} \text{T} &= \text{K} + (\text{AM}) \\ \text{K4} &= \text{K} + (\text{A4}) \\ \text{S4} &= \text{T} + (\text{A4}) \\ \text{S5} &= \text{S4} + (\text{A5}) \end{aligned}$$

Axiom (AM) says that whatever is necessary is true. The axiom can be rewritten as $\varphi \to \Diamond\varphi$, meaning that whatever is true is possible. Clearly these are desirable properties for a logic of necessity and possibility. But if we read $\Box$ as meaning "it ought to be that" or "mr. X believes that", the axiom would not make sense. In the same way the acceptability of other axioms like (A4) and (A5) depends on the meaning given to $\Box$. For this reason there is not a definitive modal logic but rather a whole family of systems build around K. As the axioms and inference rules serve to give proofs of the valid arguments of a logic, the *semantics* distinguish valid from invalid formulas by characterizing their truth behaviour in a set of designated models $\mathcal{C}$. A logic is *sound* with respect to a class $\mathcal{C}$ of models if every provable argument is valid in the semantical model, and *complete* if every valid argument has a proof in the system. A logic is *adequate* if it is both sound and complete. In propositional logic, validity can be defined using truth tables. Modal logic uses the more complex model of *possible worlds*, invented by Saul Kripke [Kri63][1]. A semantics for K could consist of the following.

[1] Actually, there were several scientists who, independently, invented similar semantics for modal logic at that time [DAFM03].

- A set W of possible worlds
- A valuation π that gives truth values (**true** or **false**) to the propositions in Φ for each of the possible worlds in W
- A binary *alternativeness relation* R (also called *accessibility relation*)

The truth value of the atomic sentence p at world w given by the valuation π may be written $\pi(p, w)$. Given this notation, the truth values of complex sentences of modal logic for a given valuation π and world $w \in W$ may be defined by the following truth clauses.

$v(\neg\varphi, w) = \textbf{true}$ iff $v(\varphi, w) = \textbf{false}$
$v(\varphi \wedge \psi, w) = \textbf{true}$ iff $v(\varphi, w) = \textbf{true}$ and $v(\psi, w) = \textbf{true}$
$v(\Box\varphi, w) = \textbf{true}$ iff for every w' in W s.t. wRw': $v(\varphi, w') = \textbf{true}$

Since $\Diamond$ is defined as $\neg\Box\neg$, we can derive from the above definition of $\Box$ that $\Diamond\varphi$ has to be read as "φ is true in *some* alternative world". A *frame* $\langle W, R\rangle$ is a pair consisting of a non-empty set W of worlds and a binary relation R on W. A *Kripke model* $\langle F, \pi\rangle$ consists of a frame F and a valuation π that assigns truth values to each atomic sentence at each world in W. An argument is *K-valid in a Kripke model M* if and only if its valuation assigns the conclusion **true** at every world in W that it assigns the premises **true**. An argument is said to be *K-valid* if and only if it is valid for every model.

It has been shown that the simplest modal logic K is adequate for K-validity. The adequateness of other model logics can be proven for certain classes of Kripke models. For instance, it can be shown that the logic T is adequate with respect to validity in models that are reflexive, the logic K4 is adequate with respect to validity in models that are transitive, the logic S4 is adequate with respect to validity in models that are reflexive and transitive and the logic S5 is adequate with respect to validity in models with accessibility relations that are equivalence relations. We will see the logic S5 again in the sections on epistemic logic and ATEL.

2.2 Temporal Logic

The term *temporal logic* has been broadly used to cover all approaches to the representation of temporal information within a logical framework, and also

more narrowly to refer specifically to the modal-logic type of approach introduced by Arthur Prior under the name of Tense Logic [Pri57, Pri67, Pri68] and subsequently developed further by logicians and computer scientists.

Temporal Logics have been applied in many areas including philosophy, linguistics, artificial intelligence and computer science. In artificial intelligence it is used to enable reasoning about action [Sho88], executable logics [Fis96] and temporal knowledge. In computer science, it serves as a foundation for *formal verification* of computer programs [Pnu77], reactive systems [MP92] as well as chip design [Var01].

There are two main kinds of temporal logics: *linear* and *branching*. In linear temporal logics, each moment in time has a unique possible future, while in branching temporal logics, each moment in time may have several possible futures. Linear temporal logic formulas are therefore interpreted over linear sequences and are regarded as specifying the behaviour of a single computation of a system. Branching temporal logics, on the other hand, are interpreted over trees describing the behaviour of the possible computations of a nondeterministic system.

The linear temporal logic LTL was developed by Pnueli in 1977 [Pnu77]. He extended classical propositional logic with the operators $\bigcirc$ (next), $\Diamond$ (sometime), $\Box$ (always) and $\mathcal{U}$ (until). The formula $\Diamond\Box p$ for example, is understood to be true if "there is a point in time after which p is always true". LTL formulas are interpreted over a single path out of the possible paths in a computation tree. To say that an LTL formula is true of a system means that the formula is true on all computation paths of the system.

The logic CTL [CE81] also uses the *path operators* $\bigcirc$, $\Diamond$, $\Box$ and $\mathcal{U}$. In addition, it has the *path quantifiers* $\exists$ and $\forall$ to quantify over one or all possible paths of the tree. For example, the CTL-formula $\forall\Box p$ is read as "in all possible computations (from now on), p will always be true", whereas $\exists\Box\, p$ is read as "in at least one computation p will always be true". An important restriction on CTL-formulas is the fact that a path operator has to always be preceded by a path quantifier. CTL without this restriction is called CTL*, which in fact subsumes both CTL and LTL.

The semantics of LTL and CTL are defined over Kripke structures where states represent moments in time, and the alternativeness relation R, often denoted as '$<$' , defines the ordering of these moments. In the case of

CTL, a state may have several successors, expressing the fact that it is not determined which course of actions a system will follow. For the formal semantics of these logics, we direct the reader to [Eme90].

An important development in the area of formal verification has been the discovery of *model checking* algorithms. Given a system or design modeled as a finite state transition graph, a *model checker* can decide whether the model satisfies a given formula. Many companies including Intel, NASA and AT&T use these techniques to shorten and improve the design phase of technologies [Var01, Vis98].

Temporal logic model checkers have been centered around these two kinds. Model checkers for the branching temporal logic CTL have proved to be very successful mainly because their running time is linear in the size of the state graph and the length of the formula [CES83]. Model checkers for the linear temporal logic LTL run in time exponential in the length of the formula, but can use a technique called 'on the fly' to require less memory than CTL model checkers [VW86].

2.3 Epistemic Logic

Epistemic logic was invented in the early 1960's by philosophers as a tool for describing epistemic concepts such as knowledge and belief formally. Their main interest was to find inherent properties of knowledge. Since then researchers from other disciplines such as linguistics, economics, game theory and computer science have become increasingly interested in reasoning about knowledge.

In the area of computer science, reasoning about knowledge plays an important role particularly in the field of intelligent agents. The relationship between an agent's knowledge and its decisions and actions has been intensively studied. Also, many of the other mental attituedes such as belief, desire, intention, obligation and permission can be modeled after the way knowledge is modeled [Duc01].

Among all approaches to epistemic logic that have been proposed, the modal approach presented by the Finnish philosopher Hintikka [Hin62] has been the most widely used for modeling knowledge. The symbol K takes the place of the necessity operator, where $K_a\varphi$ is read as "agent a knows

that φ is true". The symbol M stands for possibility, where $M_a\varphi$ is read as "agent a considers it possible that φ is true". The semantics are defined on Kripke models, where a state represents a possible 'state of affairs', more often called *possible world*. The alternativeness relation R, also called *epistemic accessibility relation*, is interpreted as indicating states that are considered possible from the agents viewpoint. For instance, if the true state of affairs is state s, and both $(s, s) \in R$ and $(s, t) \in R$, then the agent is unable to tell whether state s or t represents the true state of affairs. By R_a we denote the alternativeness relation for agent a. An agent a is said to know fact φ, or $K_a\varphi$, if in all his accessible world φ is true. $M_a\varphi$ is said to be true if in at least one accessible world φ is true. Formally, we have

$$\begin{array}{lll} s \models K_a\varphi & \text{iff} & \text{for all states } t \text{ such that } (s,t) \in R_a : t \models \varphi \\ s \models M_a\varphi & \text{iff} & \text{there exists a state } t \text{ such that } (s,t) \in R_a : t \models \varphi \end{array}$$

While K_a and M_a represent knowledge and epistemic possibility of a single agent, knowledge in groups of agents is represented by the operators E_Γ ("everybody in Γ knows" or "φ is *shared* knowledge among the members of Γ"), C_Γ ("it is common knowledge among the members of Γ") and D_Γ ("it is distributed knowledge among the members of Γ"). $E_\Gamma\varphi$ is defined to be true iff $K_a\varphi$ is true for every member of Γ. $C_\Gamma\varphi$ is defined to be true iff $E_\Gamma\varphi \wedge E_\Gamma E_\Gamma\varphi \wedge E_\Gamma E_\Gamma E_\Gamma\varphi \wedge \ldots$ is true. $D_\Gamma\varphi$ is defined to be true iff the "combined" knowledge of the members of Γ implies φ [AHK97].

As the modal operators have been given epistemic meanings, the modal axioms automatically receive new interpretations as well. The distribution axiom (AK) we gave in section 2.1 can be rewritten as

$$K_a\varphi \wedge K_a(\varphi \rightarrow \psi) \rightarrow K_a\psi$$

We see that under epistemic interpretation, (AK) now means that an agents knowledge is closed under classical logical consequence. As this is not the case with human knowledge, modal epistemic logic is often said to apply to "logically omniscient" agents, or said to apply to "implicit knowledge" of an agent [Lev84].

Depending on the desired capabilities of the agent, one may include axioms such as (AM), (A4) and (A5) described in section 2.1. (AM) $K\varphi \rightarrow \varphi$ now means that known facts are true. (A4) $K\varphi \rightarrow KK\varphi$ now means

that an agent knows that it knows something (*positive introspection*). (A5) $\neg K\varphi \rightarrow K\neg K\varphi$ (rewritten) now means that an agent knows that it does not know something (*negative introspection*). While (AM) and (A4) are rather plausible as properties of knowledge, (A5) is quite controversial [MvdH95]. However, the system S5, which includes all of these axioms, has proved to be the most practical system for knowledge among computer scientists and AI researchers. In fact, all epistemic accessibility relations we will see in the coming chapters are required to be equivalence relations.

We conclude this section with a word on belief. Modal logic of belief is sometimes called *doxastic logic*, and can be treated very similar to the logic of knowledge described above. The essential difference however is the absence of (AM). While known facts are required to be true, believed facts can be either true or false. Instead of (AM) we add the axiom

(AD) $K\varphi \rightarrow \neg K\neg\varphi$

requiring that an agent never beliefs two inconsistent facts. The system KD45, consisting of (AK), (AD), (A4) and (A5) is considered by many researchers as the standard belief logic [Duc01].

2.4 Knowledge in Distributed Systems

Throughout this thesis we will use two often used approaches to the concept of multi-agent systems[2]. The first that of a *game theorist*: to view a multi-agent system as a group of people playing a game. With the means at hand they try to win the game, or maximize their profit, or maximize their chance of winning, etc. From this point of view we try to give them one or more human like features, like knowledge, remembrance of the past, the ability to make plans or strategies, beliefs, emotions, etc. The second approach to a multi-agent system is to view it as a *distributed system* consisting of processes in a computer network running a particular protocol. It is this point of view that reminds us to take into account the technical aspects of multi-agent systems like limited memory, limited time, deadlocks, technical malfunctioning, etc. A formal model for distributed system and knowledge

[2]There is no universally accepted definition of an 'agent' or a 'multi-agent system'. See [FG97] for a recent discussion of the agent concept.

was first introduced in [HM84], and has gained wide popularity since. We will introduce it now.

We assume that, in any point of time, each of the agents is in some *state*. We refer to this as the agent's *local state*, which represents the information known to an agent, or the information encoded in the memory of a process, at that certain moment. The state of a system at a certain moment is called the *global state* and consists of each of the agents local states. If we call the local state of agent a s_a, then the global state is a tuple of the form $\langle s_1, s_2, \ldots, s_k \rangle$, where k is the number of agents. The set of all global states is called $\mathcal{G}$. Often we also define the *environment* as a factor in the system, representing everything else that is relevant in the system besides the agent's local states. One could think of hardware, user input or other agents. Usually the environment can be treated as an extra agent, which is the approach we will take in this thesis.

As agents make choices and do actions, their local states change. The system as a whole goes from one global state to the other. A sequence of global states is called a *run*. A *system* $\mathcal{R}$ *over* $\mathcal{G}$ is a set of runs over $\mathcal{G}$. If r is a run and m a moment in time, we refer to (r, m) as a *point*. We use $r(m)$ to refer to the global state in point (r, m). Furthermore, let Φ be the set of primitive propositions of the system. An *interpreted system* $\mathcal{I}$ consists of a pair (R, π), where $\mathcal{R}$ is a system over a set $\mathcal{G}$ of global states, and π is an interpretation for the propositions in Φ over $\mathcal{G}$, which assigns truth values to the primitive propositions at the global states.

We think of an agent's knowledge as being determined by its local states. In other words, it cannot distinguish between two global states in which it has the same local states, and it can distinguish between two global states in which its local state differs. Formally, if $s = \langle s_1, ..., s_k \rangle$ and $s' = \langle s'_1, ..., s'_k \rangle$ are two global states, then we say that s and s' are *indistinguishable to agent* a, and write $s \sim_a s'$ if a has the same local state in both s and s', i.e. if $s_a = s'_a$. We can extend the indistinguishability relation $\sim_a$ to points: we say that two points (r, m) and (r', m') are *indistinguishable to* a, and write $(r, m) \sim_a (r', m')$, if $r(m) \sim_a r'(m')$. Knowledge in an interpreted system $\mathcal{I}$ at a point (r, m) can now be defined similar to the original definition in section 2.3:

$$\mathcal{I}, r, m \models K_a\varphi \quad \text{iff} \quad \text{for all } (r', m') \text{ s.t. } (r, m) \sim_a (r', m') : \mathcal{I}, r', m' \models \varphi$$

Chapter 3

Alternating-time Logics

3.1 ATL

In the previous chapter we described the two main types of temporal logics, LTL and CTL. Using these logics one is able to reason about *closed systems*, systems whose behaviour is completely determined by the participating processes. Opposed to this, *open systems* are systems that interact with its environment and whose behaviour depends on the choices of the system (internal choices) as well as the behaviour of the environment (external choices). In open systems, besides universal ("do all computations satisfy a property?") and existential ("does some computation satisfy a property?") questions, a third question naturally arises: "can the system resolve its internal choices so that the satisfaction of a property φ is guaranteed no matter how the environment resolves the external choices?"

The Alternating-time Temporal Logic of Alur et al. [AHK97] addresses this question. It is based on the concept of multi-player games. The question above can be viewed as a winning condition in a two-player game between the system and the environment. A positive answer to the question corresponds with the system having a strategy guaranteeing the satisfaction of the property φ. In ATL, this is denoted by $\langle\langle system \rangle\rangle \varphi$. Generally, $\langle\langle \Gamma \rangle\rangle \varphi$ means that the coalition of agents in Γ has a joint strategy to guarantee φ, irrespective of what the other agents do. $\langle\langle \Gamma \rangle\rangle$ is actually a generalization of the path quantifiers of branching-time logic: the existential path quantifier $\exists$ corresponds to $\langle\langle \Sigma \rangle\rangle$ (Σ denoting the set of all agents), and the universal

path quantifier $\forall$ corresponds to $\langle\langle\emptyset\rangle\rangle$.

Alur et al. deal with the concept of open systems by considering the environment as one of the players in the game. Therefore, several interesting specifications for open systems, like for instance the *realizability* problem [PR89], which were usually dealt with by imposing semantic conditions on the system, can now be expressed explicitly within the logic.

Alur et al. give a symbolic *model checking* algorithm and show that the model checking problem for ATL is computationally cheap. This led to the implementation of a freely available model checker for ATL called MOCHA [AHM+98].

3.1.1 Concurrent Game Structures

ATL is based on *concurrent game structures*, which is a natural 'common denominator' for open systems[1]. A concurrent game is played on a state space. In each step of the game, every player chooses a move, and the combination of choices determines a transition from the current state to a successor state[2].

Formally, a *concurrent game structure* is a tuple $S = \langle k, Q, \Pi, \pi, d, \delta\rangle$ with the following components:

- A natural number $k \geq 1$ of *agents*.
- A finite set Q of *states*.
- A finite set Π of *propositions*.
- For each state $q \in Q$, a set $\pi(q) \subseteq \Pi$ of propositions true at q. The function π is called *valuation function*.

[1] At first, Alur, Henzinger and Kupferman used the very similar *alternating transition systems*, but changed to concurrent game structures in a second version of the paper. The reason for this might be that the system transition function in alternating transition systems is not always efficient; Jamroga gives an example of a scenario that needs only two states when modeled as a concurrent game structure, but four states when modeled as an alternating transition system [Jam03].

[2] In this thesis, we use both the words 'agent' and 'player' to denote the same concept. Which one to use is generally a matter of context, but also taste; Alur et al. solely use the term 'player', while Van de Hoek and Wooldridge solely use the term 'agent'. We will use the term 'agent' most of the time, but use the term 'player' when specificaly talking about games. In definitions we will use 'agent'.

- For each agent $a \in \{1, \ldots, k\}$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of moves available at state q to agent a. We identify the moves of agent a at state q with the numbers $1, \ldots, d_a(q)$. For each state $q \in Q$, a *move vector* at q is a tuple $\langle j_1, \ldots, j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each agent a. Given a state $q \in Q$, we write $D(q)$ for the set $\{1, \ldots, d_1(q)\} \times \ldots \times \{1, \ldots, d_k(q)\}$ of move vectors. The function D is called *move function*.

- For each state $q \in Q$ and each move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$, a state $\delta(q, j_1, \ldots, j_k) \in Q$ that results from state q if every agent $a \in \{1, \ldots, k\}$ chooses move j_a. The function δ is called *transition function*.

For two states $q, q' \in Q$ we say that q' is a *successor* of q if there is a move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$ such that $q' = \delta(q, j_1, \ldots, j_k)$. Intuitively, if q' is a successor to q, then when the system is in state q, the agents Σ can cooperate to ensure that q' is the next state the system enters.

A *computation* of S is an infinite sequence of states $\lambda = q_0, q_1, \ldots$ such that for all $u > 0$, the state q_u is a successor of q_{u-1}. Thus, a computation in a concurrent game structure is essentially the same as a run in a distributed system (see section 2.4). A computation starting in state q is referred to as a *q-computation*; if $u \in \mathbb{N}$, then we denote by $\lambda[u]$ the u'th state in λ; similarly, we denote by $\lambda[0, u]$ and $\lambda[u, \infty]$ the finite prefix $q_0, \ldots, q_u$ and the infinite suffix $q_u, q_{u+1}, \ldots$ of λ respectively.

A number of special cases of concurrent game structures are worth mentioning. In a *turn-based synchronous game structure*, at every state, only a single agent has a choice of moves. In *Moore synchronous game structures*, the state space is the product of local state spaces, one for each agent. Note that this is equivalent to the set of global states $\mathcal{G}$ in distributed systems. In every state, all agents proceed simultaneously. Each agent chooses its next local state, possibly dependent on the current local states of the other agents but independent of the moves chosen by the other agents. In *turn-based asynchronous game structures*, one agent is designated to represent a *scheduler*. In every state, the scheduler appoints one of the other agents to determine the next state.

3.1.2 Syntax and Semantics

ATL is defined with respect to a finite set Π of *propositions* and a finite set $\Sigma = \{1, \ldots, k\}$ of *agents*. An ATL formula is one of the following:

(S0) $\top$

(S1) p, where $p \in \Pi$ is a primitive proposition.

(S2) $\neg\varphi$ or $\varphi \vee \psi$, where φ and ψ are formulas of ATL.

(S3) $\langle\langle\Gamma\rangle\rangle \bigcirc \varphi$, $\langle\langle\Gamma\rangle\rangle \Box \varphi$, $\langle\langle\Gamma\rangle\rangle \varphi \mathcal{U} \psi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ and ψ are formulas of ATL.

The semantics of ATL are defined with respect to a concurrent game structure $S = \langle k, Q, \Pi, \pi, d, \delta\rangle$. First, the notion of a strategy has to be defined. A *strategy* for agent $a \in \Sigma$ is a function f_a that maps every nonempty finite state sequence $\lambda \in Q^+$ to a natural number such that if the last state of λ is q, then $f_a(\lambda) \leq d_a(q)$. Thus, the strategy f_a determines for every finite prefix λ of a computation a move $f_a(\lambda)$ for agent a. Each strategy f_a for agent a induces a set of computations that agent a can enforce. Given a state $q \in Q$, a set $\Gamma \subseteq \{1, \ldots, k\}$ of agents, and an indexed set of strategies $F_\Gamma = \{f_a \mid a \in \Gamma\}$, one for each agent $a \in \Gamma$, we define the *outcomes* of F_Γ from q to be the set $out(q, F_\Gamma)$ of q-computations that the agents in Γ enforce when they follow the strategies in F_Γ.

We write $S, q \models \varphi$ to indicate that the state q satisfies the formula φ in the structure S. The satisfaction relation $\models$ is now defined as follows:

- $S, q \models \top$
- $S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Pi$)
- $S, q \models \neg\varphi$ iff $S, q \not\models \varphi$
- $S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$
- $S, q \models \langle\langle\Gamma\rangle\rangle \bigcirc \varphi$ iff there exists a set of strategies F_Γ, one for each agent in Γ, such that for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[1] \models \varphi$
- $S, q \models \langle\langle\Gamma\rangle\rangle \Box \varphi$ iff there exists a set of strategies F_Γ, one for each agent in Γ, such that for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[u] \models \varphi$ for all $u \in \mathbb{N}$

- $S, q \models \langle\langle\Gamma\rangle\rangle \varphi \mathcal{U} \psi$ iff there exists a set of strategies F_Γ, one for each agent in Γ, such that for all $\lambda \in out(q, F_\Gamma)$, there exists some $u \in \mathbb{N}$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$

Added to this is the abbreviation $\Diamond$ (sometime), defined as $\langle\langle\Gamma\rangle\rangle \Diamond \varphi \;\hat{=}\; \langle\langle\Gamma\rangle\rangle \top \mathcal{U} \varphi$. Furthermore, the path quantifier "$[\![...]\!]\Psi$" expresses the fact that a group of agents cannot cannot cooperate to make Ψ false. Thus, one can write $[\![\Gamma]\!] \bigcirc \varphi$ for $\neg\langle\langle\Gamma\rangle\rangle \bigcirc \neg\varphi$, $[\![\Gamma]\!] \Box \varphi$ for $\neg\langle\langle\Gamma\rangle\rangle \Diamond \neg\varphi$, and similar abbreviations for $\Diamond\varphi$ and $\varphi \mathcal{U} \psi$.

Alur et al. give some scenarios to illustrate how ATL formulas might be interpreted. For instance, one can think of a train wanting to enter a railroad crossing. In order to prevent for accidents, there is a controller that can deny the train access to the crossing. The fact that the controller can prevent the train from entering the gate, can be expressed as

$$\langle\langle \mathit{controller} \rangle\rangle \Box\ \mathit{out\text{-}of\text{-}gate}$$

At the other hand, the train and the controller can cooperate so that the train will enter the gate:

$$\langle\langle \mathit{train}, \mathit{controller} \rangle\rangle \Diamond\ \mathit{in\text{-}gate}$$

3.2 ATEL

Alternating-Time Epistemic Logic [vdHW02] is an extension of ATL in which *epistemic goals* can be expressed. An epistemic goal is a goal about the knowledge possessed by an agent or group of agents. For example, Alice may have the goal of making Bob know the combination to the safe. As in ATL, agents or groups of agents may have strategies to achieve their goals. Thus, if Alice has a strategy to make Bob know the combination of the safe, which happens to be s, this would be expressed as $\langle\langle \mathit{Alice} \rangle\rangle K_B(c = s)$. Besides the individual knowledge of an agent, ATEL also covers collective forms of knowledge, like shared and common knowledge.

A model-checking algorithm for ATEL is given and shown to require polynomial time in the size of the inputs. The authors adhere to the paradigm of *planning as model checking* [GT99], in which strategies of agents can be obtained by model-checking the properties they want to achieve. In

particular, model-checking knowledge-dependent abilities is expected to generate so-called *knowledge based programs* (KBP's). These are described as programs consisting of tuples of *knowledge tests* and actions, and are a good means of describing *knowledge based protocols* [FHMV95]. The automatic generation of programs is usually refered to as *program synthesis*.

3.2.1 Syntax and Semantics

ATEL is based on *alternating epistemic transition systems* (AETS), which are an extension of the *alternating transition systems* (ATS) used by Alur et al. As previously mentioned, Alur et al. abandoned the alternating transition system in favour of the concurrent game structure. As we will follow this example and use concurrent game structues in the following chapter, we won't go in full detail about the syntax and semantics of ATEL here, but instead confine ourselves to the most important parts.

An AETS is an ATS with the addition of an *epistemic accessibility relation* $\sim_a \subseteq Q \times Q$ for each agent $a \in \Sigma$. $\sim_a$ is required to be an equivalence relation. Furthermore, $\sim_\Gamma^E$ is defined as the union of Γ's accessibility relations, or $\sim_\Gamma^E = (\bigcup_{a\in\Gamma} \sim_a)$. Also, $\sim_\Gamma^C$ denotes the transitive closure of $\sim_\Gamma^E$.

The syntax of ATEL is that of ATL extended with the epistemic operators K, E and C. It consists of the rules (S0) to (S3) (see section 3.1.2) with the addition of

(S4) $K_a\varphi$, where $a \in \Sigma$ is an agent, and φ is a formula of ATEL;

(S5) $C_\Gamma\varphi$ or $E_\Gamma\varphi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ is a formula of ATEL.

Formulas of ATEL are interpreted with respect to AETS. The definition of the satisfaction relation $\models$ is that of ATL (see section 3.1.2), with the addition of

- $S, q \models K_a\varphi$ iff for all q' such that $q \sim_a q'$: $S, q' \models \varphi$;
- $S, q \models E_\Gamma\varphi$ iff for all q' such that $q \sim_\Gamma^E q'$: $S, q' \models \varphi$;
- $S, q \models C_\Gamma\varphi$ iff for all q' such that $q \sim_\Gamma^C q'$: $S, q' \models \varphi$.

The abbreviations $\diamondsuit$ and $[\![\ldots]\!]$ are used in their original meaning.

3.2.2 Applications of ATEL

We include a few of the examples given in [vdHW02] of applications of ATEL, together with their intended meaning.

"Agent a can send private information to b, without revealing the message to another agent c":

$$K_a\varphi \wedge \neg K_b\varphi \wedge \neg K_c\varphi \wedge \langle\langle a,b\rangle\rangle \Diamond (K_a\varphi \wedge K_b\varphi \wedge \neg K_c\varphi) \tag{3.1}$$

In a simple card game, the aim of the players is to find out the deal d of cards. Having a winning strategy translates into:

$$d \rightarrow \langle\langle a\rangle\rangle \Diamond (K_a d \wedge \bigwedge_{a\neq b} \neg K_b d) \tag{3.2}$$

"If Bob knows that the combination of the safe is s, then he is able to open it (o), as long as the combination remains unchanged":

$$K_b(c=s) \rightarrow \langle\langle b\rangle\rangle (\langle\langle b\rangle\rangle \bigcirc o)\, \mathcal{U}\, \neg(c=s) \tag{3.3}$$

"The environment cannot prevent the sender from sending a message until it is received by the receiver":

$$[\![env]\!] send_m \,\mathcal{U}\, K_R m \tag{3.4}$$

"A group needs to have common knowledge of the fact that they can achieve a certain goal":

$$C_\Gamma \langle\langle \Gamma\rangle\rangle T\varphi \qquad (T \text{ a temporal operator}) \tag{3.5}$$

Indeed these ATEL formulas seem to capture useful properties about systems. Furthermore, model-checking can be used to obtain strategies by which agent a can securely communicate with agent b (3.1), player a can find out the deal of cards (3.2), etc. There is only one thing not to be overlooked: having a winning strategy does not necessarily mean the agent can play it. This we will illustrate by a game described in the following section.

3.3 The game "King Queen Ace"

In this section we introduce the game "King Queen Ace", by which we will illustrate a counter-intuitive aspect of strategies in ATEL. The game is called "King Queen Ace" since it is played with only three cards: King, Queen and Ace. The King beats the Queen, the Queen beats the Ace and the Ace beats the King. There are two players, Anna and Ben, both of whom get one card, which they don't show to the other. The third card remains on the table, face down. Then Anna has to do the only move of the game, and that is to decide to either exchange her card with the card on the table ($move_1$), or keep her card ($move_0$). The game then ends, the cards are shown and the winner is the one holding a card that beats the other's card. We will call this game "KQA-1" for short, the '1' representing the fact that only one player is allowed to move.

We model this game as a concurrent game structure, being the tuple $KQA1 = \langle k, Q, \Pi, \pi, d, \delta \rangle$, where

- $k = 2$ (Instead of the numbers 1 and 2 we will use the names Anna and Ben where possible).
- $Q = \{KQ, KA, QK, QA, AK, AQ\}$, where KQ for instance represents the state where Anna has the King and Ben has the Queen.
- Π is $\{win_{Anna}\}$, where win_{Anna} being true corresponds with Anna having won the game.
- $\pi(q) = \{win_{Anna}$ iff $q \in \{KQ, QA, AK\}$, $\emptyset$ otherwise$\}$. In the diagram below, the winning states for Anna are underlined.
- $d_A(q) = 2$, $d_B(q) = 1$ for all q (Instead of the numbers 1 and 2 we will use the names $move_0$ and $move_1$ where possible, and the name *null* for Bens 'move').
- The transition function δ is as drawn in figure 3.1.

The transition relation δ is represented by the arrows, which represent $move_1$, together with the reflexive arrows (not drawn), which represent $move_0$. For instance, $\delta(KA, \langle move_1, null \rangle) = QA$.

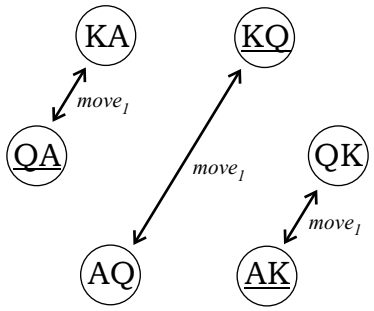

Figure 3.1: The game structure for KQA-1

Alur et al. define a strategy for agent a as a function f_a that maps every nonempty finite *state sequence* λ to a move that agent a can do in the last state of this sequence. For our game we can simply consider a strategy to be a function mapping a *state* to a move that an agent can do in that state - since there will be only one move in the game, only state sequences with length 1 will be considered.

We now ask ourselves the question: "does Anna have a winning strategy?" Or, formally spoken, does for every state q:

$$KQA1, q \models \langle\langle Anna \rangle\rangle \bigcirc win_{Anna} \qquad ? \tag{3.6}$$

In ATL, Anna does have a winning strategy:

$$f_1(q) = \begin{cases} move_0 \text{ iff } q \in \{KQ, QA, AK\} \\ move_1 \text{ otherwise} \end{cases} \tag{3.7}$$

In other words: exchange your card if the one on the table is the better one to have, otherwise keep your card. Clearly not a very appropriate strategy, since Anna has no idea if the card on the table is better or not. But in ATL, no information is hidden to any of the players. Therefore Anna can base her decisions on both her own as well as the other cards.

Since knowledge, or lack of knowledge, is the vital factor in this game, we turn to ATEL in search of a more satisfactory answer to (3.6). As we mentioned before, game structures in ATEL are similar to those in ATL, but have in addition the epistemic relation $\sim_a$. In our example, we let $\sim_{Anna}$ be the epistemic relation for Anna. We don't need to define Bens knowledge,

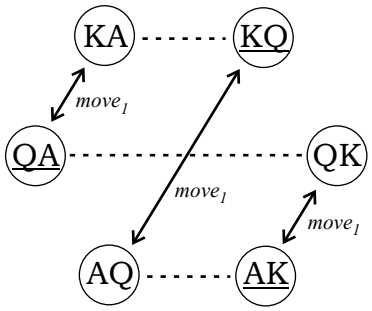

Figure 3.2: The game structure with epistemic relations for KQA-1

since it doesn't play a role in the game. $\sim_{Anna}$ is represented by the dashed lines in figure 3.2, plus reflexive arrows.

As we can see, $\sim_{Anna}$ represents the fact that Anna only knows the card she is holding. If she holds a King, she cannot distinguish between states where the other player holds a Queen or an Ace. To avoid unnecessary complication, the epistemic relation $\sim_{Anna}$ remains unchanged after Anna has made her move. One could think of Anna making her choice, but not exchanging her card yet. The game then ends and a third party decides which player has won.

The model allows us to check several possibilities, such as:

$$\langle\langle Anna\rangle\rangle \bigcirc K_{Anna} win_{Anna} \tag{3.8}$$

$$\langle\langle Anna\rangle\rangle \bigcirc (K_{Anna} win_{Anna} \vee K_{Anna} \neg win_{Anna}) \tag{3.9}$$

$$\langle\langle Anna\rangle\rangle \bigcirc \neg K_{Anna} win_{Anna} \tag{3.10}$$

Since Anna can do nothing in this game to enhance her knowledge, properties (3.8) and (3.9) are false and (3.10) is true.

A bit surprisingly, in this model (3.6) is still true. Anna still has a winning strategy, being (3.7). Although she wouldn't be able to determine whether she is in KQ or KA for instance, she is still allowed to base her strategy on which one of the two she is in. It seems to be the case that in ATEL players can have lack of knowledge, but become omniscient when it comes to strategies!

The actual meaning of (3.6) being true can better be described from the viewpoint of an external, omniscient observer who concludes that Anna has

a winning strategy, irrespective of whether she is able to play it. We are however more interested in the players viewpoint, since this is the interpretation we believe one would normally give to a question like "Does Anna have a winning strategy?"

Our game shows that, although ATEL allows for reasoning about whether an agent can bring about knowledge, it doesn't prevent agents from having a strategy which they are actually unable to play because of lack of knowledge.

3.4 Some properties reviewed

Having another look at the ATL formulas (3.1) - (3.5), we see that some of them are not expressing their intended meaning if we leave the notion of strategy as is. Proving property (3.1) in a system doesn't necessarily mean that agents a and b are now *able* to communicate secretly. Proving property (3.2) in a game isn't satisfying if the player cannot play that strategy. Property (3.3) becomes trivial, as Bob will probably have a strategy to open the safe — just dial the code s — even if he doesn't know the combination any more. He just can't put it into practice.

Because of this we have to disagree with one of the presumptions stated in [vdHW02], being the fact that model-checking a knowledge-dependent ability like (3.3) might generate a knowledge based program. Since the consequent of (3.3) is trivial, (3.3) will hold even in states where Bob doesn't know the combination, making Bobs knowledge irrelevant. Clearly, a model-checker would not be able to generate a KBP from this. What we need instead is an incorporation of knowledge into the notion of strategies. This we will do in the next chapter.

Chapter 4

Feasible ATEL

4.1 Feasible ATEL

As we have seen in the previous section, strategies in ATEL do not take into account any lack of knowledge that an agent might have, but instead assume the agents to be 'omniscient'. Because of this, many of the ATEL formulas we have seen so far did not express their intended meaning. Stronger and more realistic properties could be expressed if we only allow for strategies that agents can play based on their knowledge and lack of knowledge. In other words, an agents decisions may only be based on what he knows, not on any information hidden to him. We achieve this by imposing the following restriction on strategies: "in two states that an agent is unable to distinguish between (i.e. having the same local state), he must make the same choice." We'll call these strategies *feasible strategies* or *f-strategies*, as to distinguish them from the strategies we've seen so far. The variation of ATEL with feasible strategies we will call Feasible ATEL, or ATEL$^{\mathrm{f}}$for short.

Besides this epistemic restriction, we choose also to differ on the type of the strategy function. Whereas [vdHW02] follows [AHK97] in defining a strategy as a function from a sequence of states to a move, we define it as a function from a state to a move. This is because the first approach assumes an agent to be able to remember the history of a game, and base its strategy on it. Whether that is the case depends, in our view, on whether we choose our agents to have *perfect recall* or not. Although perfect recall

is a standard assumption made by game theorists [FHMV95], we prefer to investigate games without it first. The reasons for this can be found in section 4.5, in which we'll look into perfect recall in more detail.

4.1.1 Concurrent Epistemic Game Structures

We start by introducing the semantic structures we will use to represent our domains. These structures are an extension of the concurrent game structures used by Alur et al. to give a semantics to ATL. A *concurrent epistemic game structure* or CEGS is a tuple $S = \langle \Sigma, Q, I, \Pi, \pi, d, \delta, \sim_1, \ldots, \sim_k \rangle$ with the following components:

- A finite set $\Sigma = \{1, \ldots, k\}$ of *agents*.
- A finite set Q of *states*.
- A finite set $I \subseteq Q$ of *initial states*.
- A finite set Π of *propositions*.
- For each state $q \in Q$, a set $\pi(q) \subseteq \Pi$ of propositions true at q.
- For each agent $a \in \Sigma$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of moves available at state q to agent a. We identify the moves of agent a at state q with the numbers $1, \ldots, d_a(q)$. For each state $q \in Q$, a *move vector* at q is a tuple $\langle j_1, \ldots, j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each agent a. Given a state $q \in Q$, we write $D(q)$ for the set $\{1, \ldots, d_1(q)\} \times \ldots \times \{1, \ldots, d_k(q)\}$ of move vectors. The function D is called *move function*.
- For each state $q \in Q$ and each move vector $\langle j_1, \ldots, j_k \rangle \in D(q)$, a state $\delta(q, j_1, \ldots, j_k) \in Q$ that results from state q if every agent $a \in \Sigma$ chooses move j_a. The function δ is called *transition function*.
- For each agent $a \in \Sigma$ an *epistemic accessibility relation* $\sim_a$.

To enable the definition of feasible strategies in the next section, CEGS's need to obey the restriction that moves that are the same from the agents point of view carry the same number. Formally, for any two states $q, q' \in Q$,

agent $a \in \Sigma$ and moves $i \leq d_a(q)$ and $j \leq d_a(q')$, if $q \sim_a q'$ and i and j are the same move from the agents point of view, then $i = j$.

We use the definitions of *successor*, *computation* and *q-computation* as described in section 3.1.1.

4.1.2 Syntax en Semantics

We extend the syntax of ATEL with the operator $\langle\langle \ldots \rangle\rangle^{\mathrm{f}}$ for feasible strategies. ATEL$^{\mathrm{f}}$ is defined with respect to a finite set Π of *propositions* and a finite set $\Sigma = \{1, \ldots, k\}$ of *agents*. An ATEL$^{\mathrm{f}}$ formula is one of the following:

(S0) $\top$

(S1) p, where $p \in \Pi$ is a primitive proposition.

(S2) $\neg\varphi$ or $\varphi \vee \psi$, where φ and ψ are formulas of ATEL$^{\mathrm{f}}$.

(S3) $\langle\langle\Gamma\rangle\rangle \bigcirc \varphi$, $\langle\langle\Gamma\rangle\rangle \Box \varphi$, $\langle\langle\Gamma\rangle\rangle \varphi \mathcal{U} \psi$, $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}} \bigcirc \varphi$, $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}} \Box \varphi$, or $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}} \varphi \mathcal{U} \psi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ and ψ are formulas of ATEL$^{\mathrm{f}}$.

(S4) $K_a\varphi$, where $a \in \Sigma$ is an agent, and φ is a formula of ATEL$^{\mathrm{f}}$.

(S5) $C_\Gamma\varphi$ or $E_\Gamma\varphi$, where $\Gamma \subseteq \Sigma$ is a set of agents, and φ is a formula of ATEL$^{\mathrm{f}}$.

Additional boolean connectives are defined from $\neg$ and $\wedge$ in the usual manner. Similar to CTL, we write $\langle\langle\Gamma\rangle\rangle \Diamond \varphi$ for $\langle\langle\Gamma\rangle\rangle \mathbf{true} \mathcal{U} \varphi$.

We include the definitions of *strategy* and $out(q, F_\Gamma)$ as described in section 3.1.2. We define the shorthand $out(q, f)$ to denote $out(q, \{f\})$. Also, we define the shorthand $\bar{j}$ to denote the 'constant' strategy that plays move j in any state, or the strategy f defined as $f(q) = j$ for all q. A *feasible strategy* f_a for an agent $a \in \Sigma$ is a function $f_a : Q \to I\!N$, which must satisfy the following constraints:

- $f_a(q) \leq d_a(q) \quad$ for all $q \in Q$
- for all $q, q' \in Q$: $\quad q \sim_a q' \;\rightarrow\; f_a(q) = f_a(q')$

Note that every constant strategy $\bar{j}$ is a feasible strategy. We define a *joint strategy* F_Γ for a group of agents $\Gamma \subseteq \Sigma$ to be a set of strategies $\{f_1, \ldots, f_m\}$,

one for each agent $1, \ldots, m$ in Γ. Similarly, a *feasible joint strategy* F_Γ for a group of agents $\Gamma \subseteq \Sigma$ is a set of feasible strategies $\{f_1, \ldots, f_m\}$, one for each agent $1, \ldots, m$ in Γ.

We write $S, q \models \varphi$ to indicate that the state q satisfies the formula φ in the structure S. To reduce the length of the definition, we also use the notation $S, q, F_\Gamma \models \Psi$. The satisfaction relation $\models$ is now defined as follows:

- $S, q \models \top$
- $S, q \models p$ iff $p \in \pi(q)$ (where $p \in \Pi$).
- $S, q \models \neg\varphi$ iff $S, q \not\models \varphi$.
- $S, q \models \varphi \vee \psi$ iff $S, q \models \varphi$ or $S, q \models \psi$.
- $S, q, F_\Gamma \models \bigcirc \varphi$ iff for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[1] \models \varphi$.
- $S, q, F_\Gamma \models \Box\varphi$ iff for all $\lambda \in out(q, F_\Gamma)$, we have $S, \lambda[u] \models \varphi$ for all $u \in I\!N$.
- $S, q, F_\Gamma \models \varphi \,\mathcal{U}\, \psi$ iff for all $\lambda \in out(q, F_\Gamma)$, there exists some $u \in I\!N$ such that $S, \lambda[u] \models \psi$, and for all $0 \leq v < u$, we have $S, \lambda[v] \models \varphi$.
- $S, q \models \langle\langle\Gamma\rangle\rangle\Psi$ iff there exists a set of strategies F_Γ, one for each $a \in \Gamma$, such that $S, q, F_\Gamma \models \Psi$ (where Ψ is one of $\bigcirc\varphi$, $\Box\varphi$ or $\varphi\,\mathcal{U}\,\psi$).
- $S, q \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\Psi$ iff there exists a set of f-strategies F_Γ, one for each $a \in \Gamma$, such that $S, q, F_\Gamma \models \Psi$ (where Ψ is one of $\bigcirc\varphi$, $\Box\varphi$ or $\varphi\,\mathcal{U}\,\psi$).
- $S, q \models K_a\varphi$ iff for all q' such that $q \sim_a q'$: $S, q' \models \varphi$.
- $S, q \models E_\Gamma\varphi$ iff for all q' such that $q \sim_\Gamma^E q'$: $S, q' \models \varphi$.
- $S, q \models C_\Gamma\varphi$ iff for all q' such that $q \sim_\Gamma^C q'$: $S, q' \models \varphi$.

We include from ATL the dual operator "$[\![\ldots]\!]$" and add a second one:

$$\begin{array}{rcl} [\![\Gamma]\!]\Psi & \hat{=} & \neg\langle\langle\Gamma\rangle\rangle\neg\Psi \\ [\![\Gamma]\!]^{\mathrm{f}}\Psi & \hat{=} & \neg\langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\neg\Psi \end{array}$$

While the operator "$[\![\ldots]\!]$" expresses the fact that a group of agents cannot *ensure to avoid* some state of affairs, the operator "$[\![\ldots]\!]^{\mathrm{f}}$" expresses the fact that they cannot ensure to avoid it if they are only allowed to use feasible strategies.

The shorthand notation $S \models \langle\!\langle \Gamma \rangle\!\rangle \Psi$ holds iff there exists a set of strategies F_Γ, one for each $a \in \Gamma$, such that for all $q \in I$: $S, q, F_\Gamma \models \Psi$. Similarly, $S \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi$ holds iff there exists a set of f-strategies F_Γ, one for each $a \in \Gamma$, such that for all $q \in I$: $S, q, F_\Gamma \models \Psi$. If it is clear which system is meant, one can simply write $q \models \langle\!\langle \Gamma \rangle\!\rangle \Psi$, $q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi$, or $\langle\!\langle \Gamma \rangle\!\rangle \Psi$ or $\langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi$.

4.2 The game "King Queen Ace" again

Turning back to our game, we once more ask ourselves if Anna has a winning strategy. This time of course, we restrict ourselves to f-strategies:

$$\langle\!\langle Anna \rangle\!\rangle^{\mathrm{f}} \bigcirc win_{Anna} \quad ? \tag{4.1}$$

Strategy (3.7), repeated here, is clearly not a winning f-strategy.

$$f_1(q) = \begin{cases} move_0 & \text{iff} \quad q \in \{KQ, QA, AK\} \\ move_1 & \text{otherwise} \end{cases} \tag{4.2}$$

It violates the second restriction of f-strategies:

$$KQ \sim_{Anna} KA, \text{ but } f_1(KQ) \neq f_1(KA).$$

Intuitively, we expect that Anna doesn't have a winning feasible strategy at all. And indeed, in ATEL$^{\mathrm{f}}$, she doesn't have one. To prove this, we first need to redefine the game as a concurrent epistemic game structure. We define *KQA1* to be the tuple $\langle \Sigma, Q, I, \Pi, \pi, d, \delta, \sim_1, \ldots, \sim_k \rangle$, where Σ, Q, Π, π, d and δ are equal to those in *KQA1* defined in section 3.3, the set of initial states $I = Q$ and the epistemic relations $\sim_{Anna}$ as drawn in figure 3.2 for Anna and $\sim_{Ben}$ for Ben to be the reflexive relation[1].

Lemma 1 *KQA1* $\not\models \langle\!\langle Anna \rangle\!\rangle^{\mathrm{f}} \bigcirc win_{Anna}$

Proof:
We will prove this by contradiction. Suppose *KQA1* $\models \langle\!\langle Anna \rangle\!\rangle^{\mathrm{f}} \bigcirc win_{Anna}$.

[1] One could take $\sim_B$ to be the set depicted by the gray lines in figure 4.11 for a more realistic knowledge of Ben.

Then there exists a feasible strategy f for Anna such that for any initial state $q \in I$, $KQA1, q, f \models \bigcirc win_{Anna}$. Since $I = Q$, for any state $q \in Q$, $KQA1, q, f \models \bigcirc win_{Anna}$ holds. Then for any $q \in Q$, for any $\lambda \in out(q, f)$: $\lambda[1] \in \{KQ, QA, AK\}$. Since f is a feasible strategy, it obeys the restriction "for any $q, q' \in Q$, $q \sim_{Anna} q' \rightarrow f(q) = f(q')$". Thus, if we consider states KQ and KA for instance, since $KQ \sim_{Anna} KA$, $f(KQ)$ is equal to $f(KA)$. So either $f(KQ) = f(KA) = move_0$ or $f(KQ) = f(KA) = move_1$ holds. In the first case, if the game would start from initial state KA, then $out(KA, f)$ would give $\lambda[1] = KA$, which is contrary to what is derived above. Similarly, in the second case, if the game would start from initial state KQ, then $out(KQ, f)$ would give $\lambda[1] = KA$, which is also a contradiction. Since both of the two options prove to be contradictions, we derive $\bot$. □

4.3 Some properties reviewed again

Having another look at examples (3.1) - (3.5), we see that the following formulas express properties that are more closely resembling reality:

$$K_a\varphi \wedge \neg K_b\varphi \wedge \neg K_c\varphi \wedge \langle\!\langle a, b \rangle\!\rangle^{\mathrm{f}} \diamondsuit (K_a\varphi \wedge K_b\varphi \wedge \neg K_c\varphi) \tag{4.3}$$

Agent a and b have a way of communicating secretly, even if they don't know what other information the other agents have. They can put their strategy into practice. Note that agent a and b have a strategy to guarantee that agent c doesn't get to know φ, no matter what agent c does and no matter whether agent c's strategies are feasible or not.

$$d \rightarrow \langle\!\langle a \rangle\!\rangle^{\mathrm{f}} \diamondsuit (K_a d \wedge \bigwedge_{a \neq b} \neg K_b d) \tag{4.4}$$

Player a has a feasible strategy to get to know the deal of cards d.

$$K_{Bob}(c = s) \rightarrow \langle\!\langle Bob \rangle\!\rangle^{\mathrm{f}} (\langle\!\langle Bob \rangle\!\rangle^{\mathrm{f}} \bigcirc o) \mathcal{U} \neg(c = s) \tag{4.5}$$

Bob is able to open the safe as long as the combination doesn't change.

This last properties is an example of a knowledge precondition about which Van der Hoek and Wooldridge made the presumption: *"Model-checking*

a knowledge precondition might generate a knowledge based program". We established earlier that this presumption doesn't hold because of the fact that in ATEL, agents are omniscient in their strategies; Bob will have a strategy to open the safe even if he doesn't know the combination. With the definition of feasible strategies, agents are no longer omniscient. Their strategies are now based on their knowledge. The question arises whether the presumption above now holds. The answer is yes, but at the same time the presumption is irrelevant. This is because it is not the knowledge preconditions that are essential to a KBP, but rather the knowledge on which a feasible strategy is based. In example (4.5), it is not the fact $K_{Bob}(c = s)$ that enables him to open the safe, but rather the fact $\langle\langle Bob\rangle\rangle^{\mathrm{f}} \bigcirc o$. Bob has a feasible strategy to open the safe, and it is the knowledge on which this strategy is based that can intuitively serve as the *knowledge tests* of a KBP.

The claim we like to make is *"Model-checking a formula of the form* $S \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\varphi$ *will generate a knowledge based program."* The key insight is the fact that all the knowledge preconditions for actions that the agent has to perform are embedded in its feasible strategy. We will show how a KBP can be derived fom model-checking a formula $S \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\varphi$. Recall that a KBP is a set of tuples consisting of a knowledge test and an action. Now if the model-checker finds a formula $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\varphi$ to be true in a certain system, it has apparently found a feasible (joint) strategy F_Γ by which φ can be achieved. In the words of Giunchiglia and Traverso ([GT99]), F_Γ is a witness to the truth of the fact that φ can be guaranteed by the members of Γ. Each of the strategies in F_Γ is a function f_a that, given a state, returns a move for agent a, with the property that it returns the same move for states indistinguishable to agent a. In other words, it returns a move for any set of indistinguishable states. For each set of indistinguishable states there is a unique formula true[2]. These unique formulas are known to the agent(s) since they are true in any indistinguishable state. Therefore they can serve as the knowledge test for a KBP, enabling the agents to decide which action to perform. Thus, the KBP for agent a is the set of tuples $\{\langle\varphi_1, j_1\rangle, \ldots, \langle\varphi_k, j_k\rangle\}$, one for each set of indistinguishable states $\{S_1, \ldots, S_k\}$, where φ_i is the unique formula true in S_i and j_k the move $f_a(s)$, where s is a state of S_i.

[2]There is a unique formula true in any set of states indistinguishable to agent a if

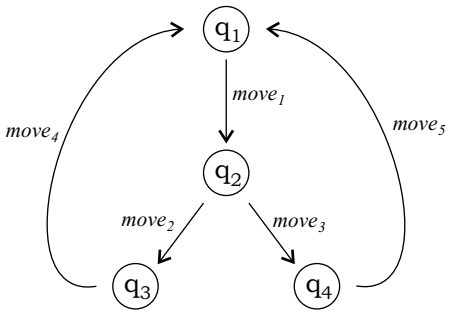

Figure 4.1: The game that led to some confusion.

4.4 A problematic case

The reason that Van der Hoek and Wooldridge didn't impose knowledge restrictions on strategies was an observation made in the game depicted in figure 4.1. No epistemic relation is defined. In the state $q1$ the agent has only one choice, leading to $q2$. In $q2$, the agent can choose between two moves, leading to either $q3$ or $q4$. The next move takes the agent back to $q1$ and from there the game continues.

The question considered is the following: "Does the agent have a strategy to visit $q3$ and $q4$ both an infinite number of times?" In ATL, it has. Strategies are based on both current and past states, so a possible ATL-strategy looks like:

$$\begin{aligned} f(q1, q2) &= q3 \\ f(q1, q2, q3, q1, q2) &= q4 \\ f(q1, q2, q3, q1, q2, q4, q1, q2) &= q3 \\ &\text{etc.} \end{aligned}$$

This strategy will lead to infinitely many visits of both $q3$ and $q4$. In ATEL, things could be different. The observation made in this model was that in

we impose the natural restriction on the epistemic accessibility relation that if an agent can distinguish between states s and t, there is at least one proposition p that has a different value in s than in t. Let S be a set of states indistinguishable to agent a. The unique formula true in S is the disjunction of, for each state $s \in S$, the conjunctions of propositions true in s and the negated propositions false in s.

ATEL, if epistemic restrictions were imposed on strategies (i.e. if strategies were feasible), the agent wouldn't have a strategy achieving infinite visits of both $q3$ and $q4$. This is because an agent should make the same choice in two states it cannot distinguish between. Since epistemic accessibility relations in ATEL are reflexive, an agent cannot 'distinguish' between $q2$ and $q2$ and it would always have to make the same choice there, leading to infinite visits of either only $q3$ or $q4$. This was against the wishes of the authors, who argued that an agent should surely be able to visit both $q3$ and $q4$ infinitely often.

We agree that it is natural to assume that an agent should have a strategy leading to an infinite number of visits of both $q3$ and $q4$. But on what basis could it do this? The agent should alternate in $q2$ between choosing $move_2$ and $move_3$, but how can it know when to choose $move_2$ and when to choose $move_3$? In other words, how could it distinguish subsequent visits of $q2$?

One possible answer is the assumption that the system is *synchronous*. Fagin et al. define synchrony as follows [FHMV95]:

> A standard assumption in many systems is that agents have acces to a shared clock, or that actions take place in rounds or steps, and agents know what round it is at all times. Put another way, it is implicitly assumed that the time is common knowledge, so that all the agents are running in synchrony.

In synchronous systems, in every state an agent knows what time it is and is therefore able to distinguish between a first and a second visit of the same state. In our example, an agent would be able to distinguish between subsequent visits of $q2$. We may expect him therefore to have a 'feasible' strategy enabling him to visit both q3 and q4 infinite times. It would look something like:

$$\begin{aligned} f(\text{the first visit of } q2) &= q3 \\ f(\text{the second visit of } q2) &= q4 \\ f(\text{the third visit of } q2) &= q3 \\ &\text{etc.} \end{aligned}$$

An other possible reason for an agent ability to distinguish between subsequent visits of $q2$ is the assumption that the agent has *perfect recall*.

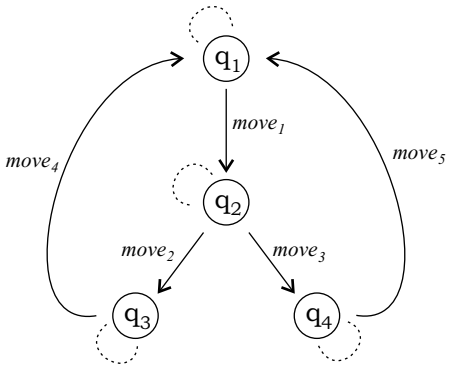

Figure 4.2: The game with reflexive epistemic relations.

This is a concept from *game theory*, meaning that an agent can remember its past, i.e. doesn't forget any observations it has done. It is easy to see that an agent with perfect recall is able to visit $q3$ and $q4$ infinite times. Such an agent could, provided that it can distinguish the four states from each other, count the number of visits to $q2$ in its past every time he is there. If he counts an even number, he chooses $move_2$, otherwise he chooses $move_3$. This will bring him an infinite number of times in both $q3$ and $q4$. Note that in ATL and ATEL, agents have perfect recall (although knowledge isn't even defined in ATEL). This is implied by the type of a strategy, being a function from a sequence of states to a move. This sequence of states represents the history of the game and since strategies in ATL and ATEL are based on this history, the agent apparantly has knowledge of this history.

The game faces us with a problem. At one hand, we would like any realistic epistemic accessibility relation to contain reflexive relations at least, including $(q2, q2)$. Otherwise, an agent in $q2$ wouldn't even consider it possible to be there. Thus, the game should at least include the epistemic relations drawn in figure 4.2. At the other hand, we would like to be able to reason about agents with perfect recall for instance. Such an agent should be able to distinguish between subsequent visits of $q2$. There is however no way in which we can express this ability in the epistemic accessibility relation while preserving reflexivity. To avoid this problem, there are two possibilities:

- Redefine the notion of strategies for agents with perfect recall. A strategy would be a function $f_a : Q^+ \rightarrow I\!N$ which takes a sequence of states as an argument and returns a move. Furthermore, one could restrict function f_a as to return the same move for state sequences that are indistinguishable from the agents' viewpoint. This approach is taken in [JvdH03], a paper that was written at the same time of this thesis. While the authors cover perfect recall, they don't address synchrony however.

- One can simply state that agents in the model in figure 4.2 cannot have perfect recall. Since in the model $q_2 \sim q_2$ holds, the agent can apparently not distinguish between subsequent visits of q_2, which implies that it doesn't have perfect recall. For the same reason, the system cannot be synchronous.

The second option is the most straight forward to choose. It leaves the definition of strategies as it is. And it is in line with an important principle observed by Fagin et al.: *perfect recall and synchrony are concepts that are expressed in the epistemic accessibility relation of an agent.* We will see in the next section how perfect recall and synchrony are defined in terms of the agents epistemic accessibility relation.

If the game in figure 4.2 forbids an agents to have perfect recall, what should we do if we do want reason about agents with perfect recall in the game in figure 4.1? The answer is *unraveling.* We should unravel the game in figure 4.1 and create a game tree where cycles are eliminated by making copies of the original states. The epistemic relation can now be defined on the game tree and obey the restrictions of perfect recall. An illustration of this we will see in the next chapter.

This approach does have a drawback however: it results in infinite models. To define an epistemic accessibility relation directly on an infinite model is impossible. To overcome this obstacle, we should not need define the epistemic relation directly on the game tree, but rather have it automatically deducted from a finite description. In the next section we suggest an approach realizing this idea, enabling us to reason about agents with perfect recall in finite models, as well as about agents in synchronous systems.

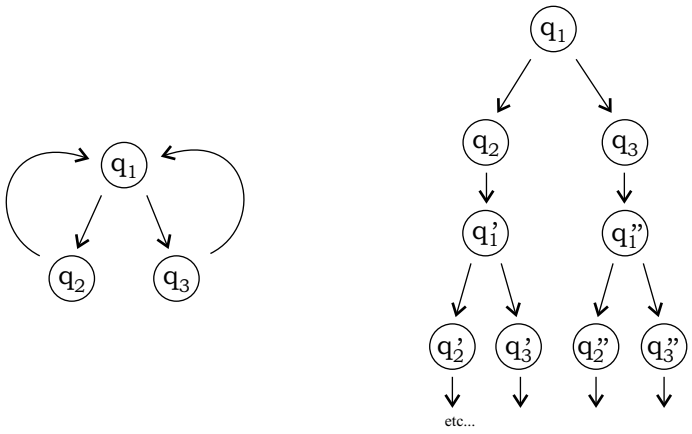

Figure 4.3: A finite interpreted environment and its corresponding game tree.

4.5 Feasible ATEL and Perfect Recall

In the previous section we saw a cyclic model for which we could not define a reflexive epistemic accessibility relation for an agent with perfect recall. We explained how the model should first be unraveled before the epistemic relations could be defined. In this section we suggest a method for reasoning about knowledge in a finite model for agents with perfect recall, which protects us from having to define the epistemic relation on an unraveled, infinite model. We will use the definition of perfect recall that Fagin et al. give for distributed systems.

Intuitively, we say that an agent has perfect recall if she is able to remember information observed or learned in the past. In other words, the agents local state encodes everything that has happened (from that agents point of view) in the run (computation) up to the current moment. The *local-state sequence of agent a at the point* (r, m) is defined as the sequence of local states she has gone through in run r up to time m. We say that *agent a has perfect recall in system* $\mathcal{R}$ if at all points (r, m) and (r', m') in $\mathcal{R}$, if $(r, m) \sim_a (r', m')$, then agent a has the same local-state sequence at both (r, m) and (r', m') [FHMV95].

To construct game structures for agents with perfect recall, we suggest

the use of an alternative semantical structure, being the *finite interpreted environment* (FIE) of Ron van de Meyden [vdMS99] - not to be confused with the 'environment' which is part of an open system. We won't go into full detail here, but merely give a sketch of our approach. An FIE is a tuple $\langle S, I, T, O, \pi, \alpha \rangle$, where S, I, T, and π are roughly similar to Q, I, δ and π in a CEGS. O is a set $(O_1, \ldots, O_k)$ of observation functions. Observations are pieces of information an agent gains in a certain state. $O_a(q)$ represents the observation agent a makes when the system is in state q. An agents knowledge is determined by the observations encoded in his local state. If the agent has perfect recall, all the observations it has done in the past are encoded in its local state. If it doesn't have any form of recall, only the observations in the current state are encoded in its local state. Figure 4.3 shows an FIE and its corresponding unraveled game tree. We will assume that the agent makes different observations in the states q_1, q_2 and q_3, and call them O_1, O_2 and O_3 respectively.

Figures 4.4 and 4.5 show the unraveled game tree and how the epistemic accessibility relation can be defined in two different ways. Figure 4.4 shows the definition of the epistemic relation for an agent without any form of recall, which we call a 'plain agent'. This agent is not able to distinguish between subsequent visits of q1, nor is it able to know in q1' whether it came via q2 or q3. Figure 4.5 shows the epistemic relation for an agent with perfect recall, which is able to distinguish between all the states in the game tree.

The examples show that by using FIE's, one can give a finite description of a game and a finite set of observations, and still be able to reason about agents with perfect recall. An additional advantage is that fact that FIE's are easy to design. This is because, in our opinion, an observation function is easier to design than an epistemic accessibility relation — the term 'observation' is more natural than 'indistinguishability'. Furthermore, once an FIE is defined, no further effort is needed for the construction of the game tree and epistemic relations, since they can be automatically derived from the FIE, given whether the agent has perfect recall or not (or whether the system is synchronous). Take for example the game tree for an agent with no form of recall (figure 4.4). In any state of the game tree, the local state of the agent consists of the observation made in that state. The observation

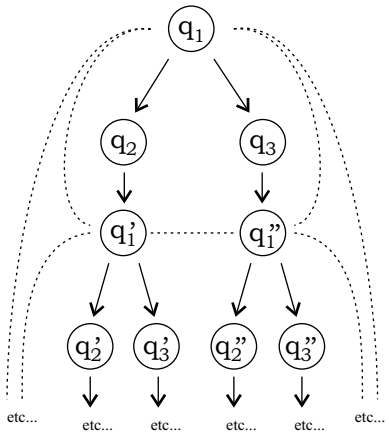

Figure 4.4: A game tree for a 'plain' agent (reflexive relations not drawn).

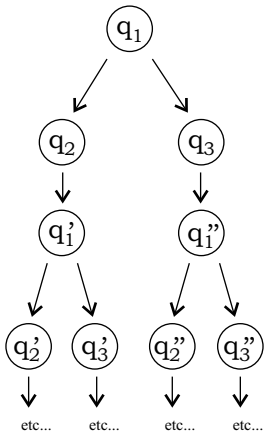

Figure 4.5: A game tree for an agent with perfect recall (reflexive relations not drawn).

made in q_1 and in q_1' is O_1 in both cases, so his local states are the same in these two states. He can therefore not distinguish between them — hence the dotted line. Now let's consider the game tree for an agent with perfect recall (figure 4.5). In any state, the local state of the agent consists of the observations done so far. Therefore, in q_1, his local state consists of O_1 only. But in q_1', his local state has grown to the set $\{O_1, O_2, O_1\}$. Thus, his local states are different in q_1 in q_1' and he is able to distinguish between them — hence the absence of a dotted line between them.

The method described above leaves us with a game tree and an epistemic accessibility relation defined on it. This game tree perfectly matches the definition of a concurrent epistemic game structure. It consists of a set of states, a valuation function, a transition function, epistemic relations for each agent and so forth. We will call the game trees CEGS's from now on. This allows us to use the notion of strategies and feasible strategies which we defined earlier.

We can for instance read from the CEGS's why an agent with perfect recall can make use of more sophisticated strategies than a plain agent. The CEGS for the plain agent in figure 4.4 has many indistinguishable states. In all of these states, a feasible strategy should select the same move. The CEGS for the agent with perfect recall however doesn't have any two states

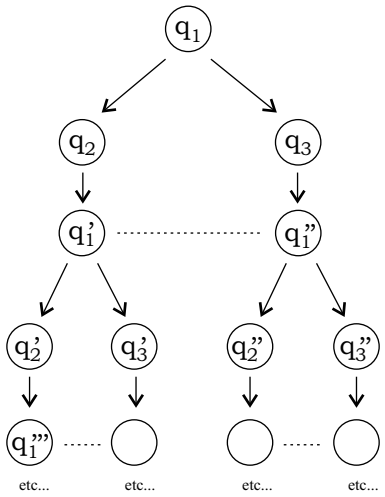

Figure 4.6: A game tree for a 'synchronous' agent (reflexive relations not drawn).

that are indistinguishable. A feasible strategy for this agent is therefore allowed to select a different move in every state. The number of different strategies the agent with perfect recall can apply is therefore much larger.

Turning back to the game in figure 4.1 once more, we see that for similar reasons the agent with perfect recall is able to visit states q_3 and q_4 an infinite number of times because he is allowed to use the strategy

$$\begin{aligned} f(q2) &= q3 \\ f(q2') &= q4 \\ f(q2'') &= q3 \\ &\text{etc.} \end{aligned}$$

The plain agent however, to whom all the states q_2, q_2', q_2'', ... are indistinguishable, is not allowed to use this strategy.

Synchrony As we mentioned in the previous section, synchrony in a system means that agents have access to a shared clock. They know what time or which round in the game it is. Formally, $\mathcal{R}$ is a *synchronous system* if for all agents a and points (r, m) and (r', m') in $\mathcal{R}$, if $(r, m) \sim_a (r', m')$, then

$m = m'$ [FHMV95]. This implies that if $m \neq m'$, then $(r, m) \not\sim_a (r', m')$. In other words, an agent can distinguish two states at different points in time. A FIE can be translated to a CEGS for agents acting synchronously using this restriction on the epistemic relation. If the agent doesn't have any form of recall, we derive his CEGS by removing from the CEGS in figure 4.4 all the 'nonhorizontal' dotted lines. The result is shown in figure in figure 4.6. We can see that synchrony implies that an agent is able to distinguish between subsequent visits of $q1$ (i.e. between $q1$, $q1'$, $q1''$, etc.), but it does not imply that he is able to distinguish between whether it came via $q2$ or $q3$.

We believe that the approach described in this section is a generic approach that allows for the automatic construction of CEGS's for agents with various characteristics. We have shown how to construct CEGS's for plain agents, agents with perfect recall and synchronous systems. We expect that other concepts can be treated along the same lines. One concept we think of is *bounded recall*; an agent with bounded recall can remember only a fixed number of most recent states in his past.

It is interesting to look at the role of perfect recall and synchrony in the context of the paradigm of 'planning as model-checking' mentioned earlier. According to this paradigm we are able to obtain strategies by model-checking. We expect that in the case of Feasible ATEL, model-checking will generate feasible strategies from which KBP's can be derived. But if we do program synthesis, we have to be aware of the relationship between properties of our agents and systems and their real-life implementations.

In this context, perfect recall is not a realistic property for an agent to have. An agent with perfect recall does not forget anything he learned in the past. Every piece of information he learns is added to the information already known. Therefore his local state must grow in order to be able to code all the new pieces of information. If a model is cyclic, there is no limit to the size to which local state could expand. Therefore agents with perfect recall for cyclic models can not be implemented — a process with perfect recall would need an infinitely large amount of memory.

Synchrony on the other hand is a much more realistic property in terms of implementation. Synchrony requires processes to act in rounds. Thus, any implementation of a synchronous system must have a mechanism to

ensure that the processes involved act synchronously. A well known protocol able to achieve this is TCP/IP [Tan96]. We believe that model-checking as a means of generating KBP's for synchronous systems is therefore an interesting option for further research.

We end this section with a word on the *opponents*. An (ATEL–)strategy is a method by which an agent can bring about, guarantee, a certain state of affairs. This means that for all possible strategies of the opponents, the agent can still guarantee its goal property. A feasible strategy is a strategy with epistemic restrictions on the moves. When we defined the meaning of an agent having a feasible strategy, we didn't impose any restrictions on the strategies of the opponents. This implies that if $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\varphi$ holds, Γ can guarantee the satisfaction of φ even if the opponents $(\Sigma\backslash\Gamma)$ are allowed to use nonfeasible strategies — even if they use 'omniscient' strategies! This is an important observation because it tells us not to worry about the mental capabilities of our opponents. If we can verify the fact that $\langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\varphi$ holds, we don't need to wonder for instance about whether the opponents have perfect recall or not; if we have a feasible strategy to achieve our goal, we are 'safe' even against omniscient opponents.

4.6 Knowing the strategy

The key feature of an f-strategy is the fact that a agent can play it based on what he knows and doesn't know. We argued that having a winning f-strategy is much more desirable than having just a winning strategy. However, having a winning f-strategy doesn't necessarily mean that we can go and collect our price. More specifically, the assertion

"agent a has a winning f-strategy"

doesn't imply either of the following assertions:

"agent a knows that he has a winning f-strategy"
"agent a knows his winning f-strategy"

This is an important observation, since the last assertion is in fact the interpretation one would give to the statement "agent a can win the game" in games of knowledge. In this section we will look at some examples for a

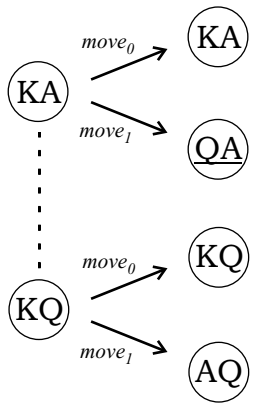

Figure 4.7: Part of the unraveled game structure for KQA-1b

better understanding of the assertions above. The examples involve formulas in individual states, which we haven't done yet. We will also show that f-strategies enable us to define the notions above formally.

We will first look to a situation where the following assertion holds:

"agent a has a winning f-strategy but doesn't know that he has it." (4.6)

Shown in figure 4.7 is part of the unraveled CEGS for the game KQA-1, but with the difference that KQ is no longer a winning state. We call this variation KQA-1b. It is easily seen that in this model, in KA, Anna has a winning f-strategy, being $\overline{move}_1$. But she considers KQ a possible world, and in KQ she doesn't have a winning f-strategy. Therefore in KA, she cannot know that she has a winning f-strategy. Formally, we can express this situation as follows:

$$KA \models \langle\langle Anna \rangle\rangle^{\mathrm{f}} \bigcirc win_{Anna} \wedge \neg K_{Anna} \langle\langle Anna \rangle\rangle^{\mathrm{f}} \bigcirc win_{Anna}$$

Note that the f-strategy $\overline{move}_1$ is a normal strategy as well (i.e. not necessarily feasible). In fact, all 'one step' f-strategies are normal strategies. The reader be not distracted by this detail, since our example merely serves to give an instance of (4.6).

Secondly, we look at the following situation:

"agent a has a winning f-strategy, he knows that he has one, but he doesn't know which one it is." (4.7)

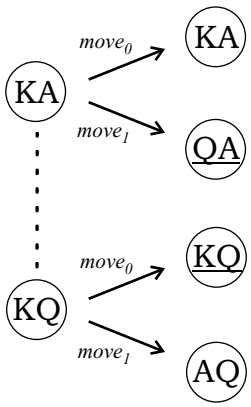

Figure 4.8: Part of the unraveled game structure for KQA-1.

We look at the same CEGS, but with KQ a winning state again, as in KQA-1. This time, Anna has a winning f-strategy both in KA and KQ, being $\overline{move}_1$ and $\overline{move}_0$ respectively. Therefore she knows that she has a winning f-strategy. But in KA, Anna is unable to know which of the two strategies will let her win the game. This is because she doesn't know whether she is in KA, in which case she needs to play $\overline{move}_1$, or in KQ, in which case she needs to play $\overline{move}_0$. We say that, although Anna knows that she has a winning f-strategy, she doesn't know its identity.

Giving a formal representation for this situation is not as straightforward this time. This is because our language doesn't have any means of identifying strategies. It is however possible to define 'knowing a strategy' if we make use of the fact that our agents are intelligent. They know the game and are aware of all the possible situations and moves. Therefore they are able to construct the CEGS's in their 'minds'. This enables them to reason about the game, about possible outcomes of moves and strategies and about their knowledge, as well as that of others. Therefore, if there is a strategy that leads to a desired outcome in all equivalent states, the agent is aware of its existence, is able to identify this strategy and therefore knows it. In exactly the same way an agent is able to know a f-strategy leading to a desired outcome. This principle can be more formally stated as follows:

in system S, in state q, agent a knows f-strategy f_a leading to

Ψ iff f_a is an f-strategy for agent a, such that for all $q' \sim_a q$, $S, q', f_a \models \Psi$

We believe that 'knowing your winning f-strategy' is actually the mental state a player should achieve in a knowledge game to be sure to win. Therefore we will introduce a new operator for this in the next section, in which we will also investigate how this translates to f-strategies for coalitions.

We have shown that a player can have an f-strategy in a certain state without knowing how to win. The question arises whether it is necessary to reason about individual states. Couldn't we just stick to general formulas like (4.1), which require a winning strategy for all initial states, to decide whether a player can win the game or not? Indeed, in our game KQA-1 this gives the desired result: (4.1) does not hold, Anna cannot win the game. But in many other games it is very interesting to reason about states (and thus subgames) individually. This is the case for instance when a player can't win the game for sure from any initial situation, but might find himself in a situation later on where he can. We give as an example the game KQA-1c, with different winning conditions again: Anna wins when she holds the King, otherwise she loses (figure 4.9). In this game, Anna doesn't have a winning (f-)strategy in general. Once the cards are dealt, the situation might be different however. If the cards are dealt QA or AQ, Anna does have a winning f-strategy, but doesn't know this for sure[3]. More importantly, if the cards are dealt KA or KQ, Anna has a winning strategy, $\overline{\overline{move}}_0$ and knows it as well. This example shows that knowledge about a winning f-strategy, and thus being in a winning position, might occur *during* a game.

4.7 Knowledge of strategies in groups

The previous section shows that in order to be sure to win, an agent needs a winning f-strategy and needs to know its identity. We gave a formal definition for 'knowing a strategy' in the case of a single agent. In the case of more than one agent, 'knowing a strategy' can mean several things. This is because knowledge in groups comes in different flavours. We introduced

[3]She would of course play strategy $\overline{\overline{move}}_1$, since that might make her win the game, but we leave this observation for what it is.

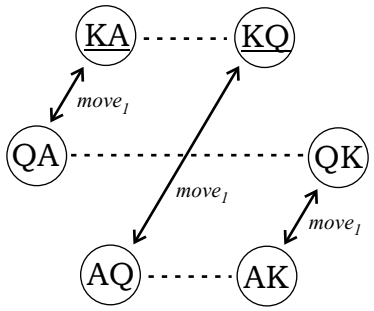

Figure 4.9: The game structure for the game KQA-1c.

in section 2.3 the operators for collective knowledge $C_\Gamma\varphi$, $E_\Gamma\varphi$ and $D_\Gamma\varphi$. As we will show, knowledge of (joint) strategies in ATEL$^{\mathrm{f}}$ can be in one of these forms as well. We extend ATEL$^{\mathrm{f}}$ to be able to express these types of knowledge and give examples to illustrate them.

To be able to provide illustrations of collective knowledge of strategies, we introduce a variation of KQA: KQA-2. In this game, both players receive a random card, as they did in KQA. But this time both players are allowed to swap their card with the card on the table. First Anna, then Ben. If both players choose to swap, Ben will receive the card that Anna was holding. The aim for the players is to work together and win the game. We will vary the winning states to produce three variants of KQA-2. As the game model is a bit more complicated now, we show the possible transitions and epistemic relations in separate figures. The transitions are shown in figure 4.10. The black arrows represent the possible moves of Anna and the gray arrows those of Ben. The reflexive arrows are not drawn. We'll call the move of exchanging the card $move_1$ and the move of keeping the card $move_0$. As Anna is first allowed to make a move, followed by Ben, examples of possible state sequences are: KA, QA, QK or KA, KA, KQ or KQ, KQ, KQ.

The epistemic accessibility relations are shown in figure 4.11. The black lines represent the knowledge of Anna, the gray lines that of Ben. Again the reflexive lines are not drawn. The epistemic accessibility relations show that Anna cannot distinguish between KA and KQ, and similarly Ben cannot distinguish between KA and QA. Another kind of information that can be observed from this model by following the lines is for instance the follow-

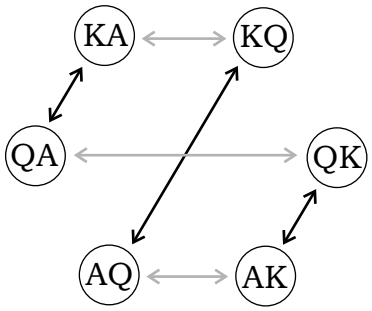

Figure 4.10: The transitions of KQA-2.

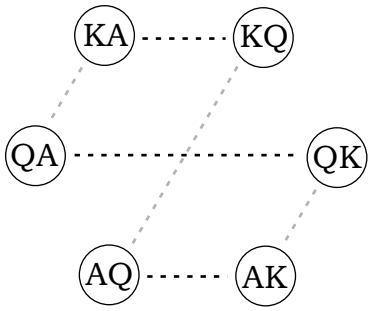

Figure 4.11: The epistemic accessibility relations of KQA-2.

ing: if the cards are dealt KQ, Anna considers KA possible, and therefore considers it possible that Ben considers QA possible. In other words, Anna considers it possible that Ben considers it possible that Anna has the Queen.

KQA-2a: *KQ* and *QK* The first variant of KQA-2 we will look at is called KQA-2a and has winning states KQ and QK. In other words, in order for Anna and Ben to win the game, they must try to get hold of the King and the Queen. Now suppose the cards are dealt KA. The obvious strategy for Anna would be $\overline{move}_0$, while Ben should play $\overline{move}_1$. Anna will hold on to her King, and she is sure that Ben will be able to get hold of the Queen. Ben initially sees that he has an Ace. He knows that Anna has got either the King or the Queen, which she should keep, after which he can get

hold of either the Queen or the King. Clearly our coalition is in a winning position. Let's consider the joint strategy

$$F_1 = \{\overline{\overline{move}}_0, \overline{\overline{move}}_1\}$$

This will make the game will end in KQ and is therefore a winning strategy. It's also an f-strategy since both players have a fixed move to play (i.e. their move is not depending on what they know). It is however not common knowledge among the players. Anna can imagine that Ben has a Queen. In that case, F_1 would make the game end in KA, and the players would lose. In other words, Anna can imagine that Ben doesn't know that F_1 is a winning strategy.

Now let's consider the joint strategy

$$\begin{aligned} F_2 = \{&\{AQ, AK \to move_1,\ move_0 \text{ otherwise}\},\\ &\{KA, QA \to move_1,\ move_0 \text{ otherwise}\}\} \end{aligned}$$

In this strategy the players only swap cards if they hold the Ace. Again this is a winning strategy since the game will end up in KQ. It is an f-strategy since the strategies of the players are based on their knowledge, namely which card they are holding. Moreover, the fact that F_2 is a winning strategy is common knowledge among the players, since F_2 is a winning strategy from every state in the model. To capture the concept of a winning strategy for a group of players that is common knowledge among them, we define the following operator:

- $S, q \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_C\ \Psi$ iff there exists a feasible joint strategy F_Γ for the group of agents Γ, such that for all $q' \sim_C q$ it holds that $S, q', F_\Gamma \models \Psi$

KQA-2b: *KQ* In the game KQA-2b the only winning state is KQ. Suppose the cards are dealt KA again. Consider the following strategy

$$\begin{aligned} F_3 = \{&\{QK, QA, AK, AQ \to move_1,\ move_0 \text{ otherwise}\},\\ &\{KA, QA, AK, QK \to move_1,\ move_0 \text{ otherwise}\}\} \end{aligned}$$

In other words, exchange your card if it not the one you need. In this game, F_3 is a winning f-strategy in KA. It is not common knowledge, since the

game cannot be won from states AK and QK. It is true however that both player know that it is a winning strategy, since F_3 is a winning strategy from the states Anna and Ben consider possible, i.e. KA, KQ and QA. In other words, everybody knows that F_3 is a winning strategy. For shared knowledge of a strategy we define the following operator:

- $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E}\ \Psi$ iff there exists a feasible joint strategy F_Γ for the group of agents Γ, such that for all $q' \sim_E q$ it holds that $S, q', F_\Gamma \models \Psi$

KQA-2c: *QK* In the game KQA-2c the only winning state is QK. Suppose the cards are dealt KA again. Consider the following strategy

$$F_4 = \{\{KQ, KA, AK, AQ \rightarrow move_1,\ move_0 \text{ otherwise}\},\\ \{KQ, AQ, KA, QA \rightarrow move_1,\ move_0 \text{ otherwise}\}\}$$

Like F_3, F_4 exchanges the card if it not the one you need. F_4 is a winning feasible joint strategy. This is not common knowledge. Nor is it shared knowledge, since Anna considers KQ a possible world, from where the game cannot be won. Ben however knows that F_4 is a winning strategy, since this is the case in both KA and QA. Anna only considers it possible that F_4 is a winning joint strategy. For individual knowledge of a strategy we define the following operators:

- $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a}\ \Psi$ iff there exists a feasible joint strategy F_Γ for the group of agents Γ, such that for all $q' \sim_a q$ it holds that $S, q', F_\Gamma \models \Psi$
- $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{M_a}\ \Psi$ iff there exists a feasible joint strategy F_Γ for the group of agents Γ, such that there exists a state $q' \sim_a q$, such that $S, q', F_\Gamma \models \Psi$

Note that $\langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{K_a}$ defines the concept of "knowing your winning strategy" for the case of a single agent, which we described in the previous section.

To show the relations between the different operators, we proof the following lemma's.

Lemma 2 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C}\ \Psi \quad \Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E}\ \Psi.$

Proof: Rewriting both sides gives:

there exists an f-strategy F_Γ such that for all $q' \sim_\Gamma^C q$ it holds that $S, q', F_\Gamma \models \Psi$ $\Rightarrow$ there exists an f-strategy F_Γ such that for all $q' \sim_\Gamma^E q$ it holds that $S, q', F_\Gamma \models \Psi$

Since $\sim_\Gamma^C$ is the transitive closure of $\sim_\Gamma^E$, it is easy to see that the lemma holds. □

Lemma 3 $S, q \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_E \Psi \quad \Rightarrow \quad S, q \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_{K_a} \Psi$ (for $a \in \Gamma$).

Proof: Rewriting both sides gives:

there exists an f-strategy F_Γ such that for all $q' \sim_\Gamma^E q$ it holds that $S, q', F_\Gamma \models \Psi$ $\Rightarrow$ there exists an f-strategy F_Γ such that for all $q' \sim_a q$ it holds that $S, q', F_\Gamma \models \Psi$

Since $\sim_\Gamma^E$ is the union of $\sim_a$ for all $a \in \Gamma$, it is easy to see that the lemma holds. □

Lemma 4 $S, q \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_C \Psi \quad \Rightarrow \quad S, q \models C_\Gamma \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_C \Psi$.

Proof: Rewriting the left side gives

there exists an f-strategy F_Γ such that for all $q' \sim_\Gamma^C q$ it holds that $S, q', F_\Gamma \models \Psi$

Rewriting the right side gives

for all $q' \sim_\Gamma^C q$ it holds that $S, q' \models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_C \Psi$

which can be rewritten to

for all $q' \sim_\Gamma^C q$ there exists an f-strategy F'_Γ such that for all $q'' \sim_\Gamma^C q'$ it holds that $S, q'', F'_\Gamma \models \Psi$

Because of the tautology $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$ we may rewrite this to obtain

for all $q' \sim_\Gamma^C q$, for all $q'' \sim_\Gamma^C q'$, there exists an f-strategy F'_Γ such that $S, q'', F'_\Gamma \models \Psi$

Because $\sim_\Gamma^C$ is closed under transitivity, thus transitive, we rewrite to

for all $q'' \sim_\Gamma^C q$, there exists an f-strategy F'_Γ such that $S, q'', F'_\Gamma \models \Psi$

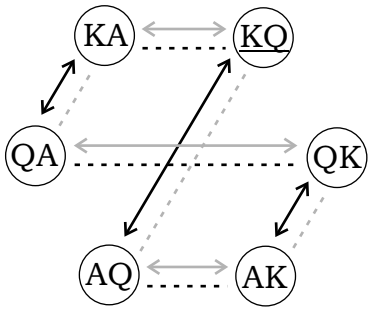

Figure 4.12: The game structure for KQA-2b.

Thus, we have rewritten the lemma to

> there exists an f-strategy F_Γ such that for all $q' \sim_\Gamma^C q$ it holds that $S, q', F_\Gamma \models \Psi \quad \Rightarrow \quad$ for all $q'' \sim_\Gamma^C q$, there exists an f-strategy F'_Γ such that $S, q'', F'_\Gamma \models \Psi$

which is again an instance of the tautology $\exists x \forall y P(x,y) \Rightarrow \forall y \exists x P(x,y)$. □

Lemma 5 *It is not the case that for all S, q, Γ and Ψ, it holds that*

$$S, q \models \langle\langle \Gamma \rangle\rangle_E^{\mathrm{f}} \Psi \quad \Rightarrow \quad S, q \models E_\Gamma \langle\langle \Gamma \rangle\rangle_E^{\mathrm{f}} \Psi$$

Proof: We will construct a counterexample that satisfies $S, q \models \langle\langle \Gamma \rangle\rangle_E^{\mathrm{f}}$ but not $S, q \models E_\Gamma \langle\langle \Gamma \rangle\rangle_E^{\mathrm{f}} \Psi$. We make use of the game KQA-2b, shown in figure 4.12. In KQA-2b, the only winning state for Anna and Ben is KQ. Anna and Ben have a winning joint strategy in the states KA, KQ, QA and AQ, but not in QK and AK. Informally spoken, we will show that Ben doesn't know in KA of a feasible joint strategy that is shared knowledge among Anna and him. Formally, we show

$$KA \models \langle\langle Anna, Ben \rangle\rangle_E^{\mathrm{f}} \bigdiamond win \quad \text{and} \quad KA \not\models E_{Anne,Ben} \langle\langle Anne, Ben \rangle\rangle_E^{\mathrm{f}} \bigdiamond win$$

We use $\sim_E$ to denote $\sim_{\{Anne,Ben\}}^E$. The left side holds if there exists an f-strategy F such that for all $q' \sim_E KA$ it holds that $S, q', F \models \bigdiamond win$ (where S is the CEGS for KQA-2b). This assertion holds if there exists an f-strategy F such that for all $q' \in \{KA, KQ, QA\}$ it holds that $S, q', F \models \bigdiamond win$. The strategy F_3 defined in section 4.7 satisfies this requirement.

To prove the right side, we have to show that there exists a state $q' \sim_E KA$ for which there doesn't exists an f-strategy F such that for all $q'' \sim_E q'$ it holds that $S, q'', F \models \bigdiamondsuit win$. We choose q' to be the state QA and have to show that there doesn't exists an f-strategy F such that for all $q'' \sim_E QA$ it holds that $S, q'', F \models \bigdiamondsuit win$. Therefore we have to show that there doesn't exists an f-strategy F such that for all $q'' \in \{KA, QA, QK\}$ it holds that $S, q'', F \models \bigdiamondsuit win$. For this it suffices to prove that there doesn't exists an f-strategy F such that $S, QK, F \models \bigdiamondsuit win$. This was given above. □

Lemma 6 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E}\ \Psi \quad \Rightarrow \quad S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi.$

Proof: Rewriting both sides gives:

> there exists a f-strategy F_Γ such that for all $q' \sim^E_\Gamma q$ it holds that $S, q', F_\Gamma \models \Psi \quad \Rightarrow \quad$ for all $q' \sim^E_\Gamma q$ it holds that $S, q' \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\, \Psi$

which gives:

> there exists a f-strategy F_Γ such that for all $q' \sim^E_\Gamma q$ it holds that $S, q', F_\Gamma \models \Psi \quad \Rightarrow \quad$ for all $q' \sim^E_\Gamma q$ there exists an f-strategy F_Γ such that $S, q', F_\Gamma \models \Psi$

which is an instance of the tautology $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$. □

Lemma 7 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a}\ \Psi \quad \Rightarrow \quad S, q \models K_a \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a}\ \Psi.$

Proof: The proof is exactly the same as the proof for lemma 4, with $\sim_a$ substituted for $\sim^C_\Gamma$, K_a substituted for C and the observation that $\sim_a$ is transitive. □

Lemma 8

$$S, q \models C_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C}\ \Psi \qquad \textit{and}$$
$$S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E}\ \Psi \qquad \textit{and}$$
$$S, q \models K_a \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a}\ \Psi$$

Proof: We will prove these three assertions by constructing a counterexample which satisfies

$$S, q \models C_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi \quad \text{and not} \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a}\ \Psi$$

From this counterexample, the three assertions can be easily derived. This is because $S, q \models C_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi$ implies $S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}\ \Psi$ (lemma 2) and $S, q \models$

$K_a\langle\langle\Gamma\rangle\rangle^{\mathrm{f}}\,\Psi$ (lemma 3), and because the implication $S,q \not\models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_{K_a}\,\Psi \Rightarrow S,q \not\models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_{E}\,\Psi \Rightarrow S,q \not\models \langle\langle\Gamma\rangle\rangle^{\mathrm{f}}_{C}\,\Psi$ is trivial. The example is taken from KQA-1 (see figure 4.8).

$$\begin{aligned}\text{Claim:}\quad & KQA1, KA \models C_{Anna}\langle\langle Anna\rangle\rangle^{\mathrm{f}} \bigcirc win_{Anna} \quad \text{and}\\ & KQA1, KA \not\models \langle\langle Anna\rangle\rangle^{\mathrm{f}}_{K_{Anna}} \bigcirc win_{Anna}\end{aligned}$$

Since the common knowledge relation C_{Anna} equals to K_{Anna}, we have to show that

for KA and KQ there exists f-strategies f_1 and f_2 such that $KQA1, KA, f_1 \models win_{Anna}$ and $KQA1, KQ, f_2 \models win_{Anna}$

and

there doesn't exists an f-strategy f_3 such that it holds that $KQA1, KA, f_3 \models win_{Anna}$ and $KQA1, KQ, f_3 \models win_{Anna}$

f_1 and f_2 are easily found:

$$\begin{aligned}f_1 &= \overline{\overline{move}}_1\\ f_2 &= \overline{\overline{move}}_0\end{aligned}$$

To show that f_3 doesn't exist, we suppose that it does. Since f_3 is a feasible strategy, it has to select the same moves in KA and KQ. So f_3 has to select $move_0$ in both KA and KQ or $move_1$ in both of them. In both cases, f_3 clearly doesn't satisfy the necessary conditions:

$KQA1, KA, f_3 \models win_{Anna}$ and $KQA1, KQ, f_3 \models win_{Anna}$ □

The operators $\langle\langle\ldots\rangle\rangle^{\mathrm{f}}_{C}$, $\langle\langle\ldots\rangle\rangle^{\mathrm{f}}_{E}$, $\langle\langle\ldots\rangle\rangle^{\mathrm{f}}_{K_a}$, and $\langle\langle\ldots\rangle\rangle^{\mathrm{f}}_{M_a}$ can be very useful as tools for discussing and defining various notions that arise in scenarios involving strategies and knowledge for groups of agents. As an example, consider the scenario where Anna and Ben have a feasible joint strategy that is shared knowledge among them, but not common knowledge. Although they both have the same winning strategy in mind, they might not know whether they are actually going to win the game, since they might consider it possible that the other one doesn't know the strategy. Another interesting example is the following: consider a group of agents that have a feasible joint strategy that is common knowledge among all of them except one[4].

This one agent merely considers the strategy possible ($\langle\!\langle \ldots \rangle\!\rangle^{\mathrm{f}}_{M_a}$). One can speculate about the chances these agents have. Suppose the strategy this one agent considers possible is the *only one* he considers possible. We can expect him to play that strategy then, since it knows of no other way to win the game. Suppose the other agents know that this agent only holds this strategy possible. Then they might anticipate on the fact that he is going to play it. It would be sensible if they play the strategy as well, since that is *their* only chance of winning now. If they expect this from each other as well, they might even conclude that they are in a *winning position*, i.e. sure that they are going to win.

These are only two of many interesting scenarios worth considering. They lead us to an interesting question: when is a coalition in a winning position? In this thesis, we won't attempt to fully answer this question but confine ourselves to an important observation in the following section.

4.8 Winning position

In section 4.6 we claimed that 'knowing your winning strategy' is the state an agent should achieve in able to win the game. We use the term *winning position* to describe the situation where a single agent or a group of agents "is sure that they are going to win". We use the word 'win' because most of the examples we consider are games. In general, a winning position can be thought of as the situation where a group of agents "is sure that they are going to guarantee the satisfaction of a certain property φ.

In the previous section we have shown that knowledge of strategies in *groups* of agents can be in different forms. The question arises which mental state a group of agents should have to be in a winning position. Would common knowledge of a winning strategy for instance be enough? The answer is negative. Common knowledge of a winning strategy is to weak a demand for a group to be sure that they are going to win. We will show this using a well known game in game theory, "Matching Pennies".

In the game Matching Pennies (MP) two players, Anna and Ben, have

[4]Note that to actually express this, we would need to define a new operator $\langle\!\langle \Gamma \rangle\!\rangle^{f}_{C_\Lambda}$ denoting the fact that a group Γ has a strategy that is common knowledge among the members of group Λ.

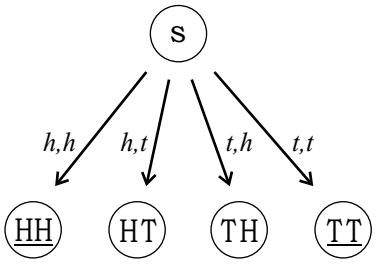

Figure 4.13: The game structure for "Matching Pennies".

to choose a side of a coin. If they both choose heads or both choose tails they win, otherwise they loose. Figure 4.13 shows the game model. The players' choices are written besides the arrows; 'h, t' for instance means that Anna chooses head and Ben chooses tail. This choice will bring them in the state HT, which is not a winning state. Their knowledge is defined by the set of reflexive arrows. It is easily seen that in s, the initial state, the players have two winning joint strategies: $\{\bar{h}, \bar{h}\}$ (both choose head) and $\{\bar{t}, \bar{t}\}$ (both choose tail). Both these strategies are common knowledge among Anna and Ben, since they are winning strategies from every state in the closure of $\{s\}$ under $\sim_C$, which is $\{s\}$. Since the strategies are feasible as well (remember that all one-step strategies are feasible), we can conclude

$$MP, s \models \langle\langle Anna, Ben \rangle\rangle^{\mathrm{f}}_C \; win$$

However, Anne and Ben are not in a winning position. They both have to choose and play one of the two strategies but they don't know which one the other is going to choose. Only if they make the same choice will they win, otherwise they will lose. Since they do not know which strategy the other is going to choose, they can not be sure that they are going to win.

The reason for this dilemma is the existence of more than one possible strategy that is common knowledge among the agents. If there would be only one, the agents didn't have to think about which one to use. They would use this one and know that the others would use it as well. In this case they are in a winning position. If there are more strategies that are common knowledge, the agents don't know which one to use. Unless however, we ascribe to the agents the ability of being able to 'sort' their strategies, lexicographic for

instance. Then they could sort their strategies and use the strategy that comes first in the ordering. This would be the same one for every agent, therefore they know which one everybody is going to use and they are in a winning position again. In fact, we could do with a slightly less strong demand then sorting. If a group of agents has a uniform selection method, i.e. an method that, given a set of strategies, selects the same strategy for every agent, they are already in a winning position. Formally, we define a *uniform strategy-selection method* for a group of agents Γ to be a function $u_\Gamma : (\Sigma, 2^\Omega) \to \Omega$, where Ω is the set of all strategies, which must satisfy the constraint that if $H \subseteq \Omega$ is a set of strategies, $u_\Gamma(a, H) = u_\Gamma(a', H)$ for any two agents a and a' from Γ.

Lemma 9 *If Γ is a group of agents with a uniform strategy-selection method u_Γ and $r \models \varphi$ iff r is a winning state, and it is common knowledge among the agents 1. which states are the winning states, 2. that every agent will use u_Γ to select a strategy and play it, then*

$$q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_C \varphi \quad \Longrightarrow \quad \Gamma \textit{ is in a winning position in state } q$$

Proof: If $q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_C \varphi$, then there exists a feasible joint strategy F_Γ for the group of agents Γ, such that for all $q' \sim_C q$ it holds that $S, q', F_\Gamma \models \Psi$. In other words, there exists a set H of feasible joint strategies that are common knowledge among the agents, with $|H| \geq 1$. Since the agents have a uniform selection method u_Γ, every agent will select from H the same strategy $u_\Gamma(H) = G_\Gamma$. Remember that G_Γ is a set of feasible strategies $\{g_1, \ldots, g_m\}$, one for every agent $1, \ldots, m$ in Γ. Every agent a will play his personal strategy g_a from the joint strategy G_Γ, and knows that the other agents will do so too. Therefore, every agent knows that G_Γ will be executed and therefore that φ will be guaranteed. In other words, every agent knows that they are going to win. □

From the lemma above and with the help of an extra lemma, a winning position for a single agent can be easily derived.

Lemma 10 $q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{K_a} \varphi \iff q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_E \varphi \iff q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_C \varphi$

Proof: The implications from right to left we have from lemma's 2 and 3. To prove the implications from left to right, we have to prove that if there

exists a feasible strategy f_a for agent a, such that for all $q' \sim_a q$ it holds that $S, q', F_\Gamma \models \Psi$, we have that $S, q', F_\Gamma \models \Psi$ holds for all $q' \sim_E q$ and for all all $q' \sim_C q$ (with respect to the group of agents $\{a\}$). For this it suffices to prove that if $q' \sim_a q$, then $q' \sim_E q$ and $q' \sim_C q$ (with respect to the group of agents $\{a\}$). For this it suffices to prove that the relation $\sim_a$ is equivalent to $\sim_E$ and $\sim_C$ (with respect to the group of agents $\{a\}$). $\sim_E$ is defined as the union of the set $\{\sim_a\}$, thus equivalent. $\sim_C$ is defined as the transitive closure of $\sim_E$ and therefore of $\sim_a$. Since $\sim_a$ is transitive already, $\sim_C$ is equivalent to $\sim_a$. □

Lemma 11 *If $q \models \varphi$ iff q is a winning state, then*

$$q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{K_a} \varphi \quad \Longrightarrow \quad \textit{agent } a \textit{ is in a winning position in state } q$$

Proof: Directly from lemma's 9 and 10. □

Conclusion

Alternating-time Temporal Epistemic Logic is a powerful logic to reason about time and strategies for multi agent environments, as is ATL. In addition, ATEL allows for reasoning about various epistemic notions for agents with incomplete knowledge. Besides the knowledge of agents and groups of agents, strategies attaining knowledge and knowledge of having a strategy can be expressed in ATEL.

The definition of strategies in ATEL was taken from ATL and left untouched. This led us to conclude that strategies in ATEL are complete information strategies; the agents are not hindered by any lack of knowledge when choosing their actions but instead have full information about the state of the game, at any time. We identified this as a counterintuitive feature.

In Feasible ATEL, we altered the definition of strategies in order to account for the fact that agents can have incomplete information when choosing their next move. This was done by demanding their choices to be the same for states they cannot distinguish from each other. This amounts to saying that their strategy should be based on their local state only. By use of examples we showed that strategies in Feasible ATEL indeed enable the modeling of realistic scenarios. We also examined a slightly controversial scenario and showed that the possibilities of an agent by means of its strategy depend on several factors; We discussed perfect recall and synchrony, compared them to the 'plain' agent and looked their implications in real-life implementations.

We then showed that having a feasible strategy doesn't necessarily imply that the agent is able to enforce it. It is possible that the agent doesn't know its feasible strategy or that it knows it has one but does not know

which one. We identified the need for an agent or for a coalition to know the identity of its feasible strategy. We formalized this using the operators $\langle\!\langle \ldots \rangle\!\rangle^{\mathrm{f}}_{C}$, $\langle\!\langle \ldots \rangle\!\rangle^{\mathrm{f}}_{E}$ and $\langle\!\langle \ldots \rangle\!\rangle^{\mathrm{f}}_{K_{\ldots}}$ for common knowledge, shared knowledge and individual knowledge respectively. Their mutual relations were made clear with the help of a number of lemma's.

We argued that the state a group of agents is actually required to achieve is that of being in a winning position. We defined a group of agents being in a winning position as a group of agents knowing that they are going to win. We established that common knowledge of a winning feasible strategy is not a sufficient requirement for a group of agents to be in a winning position. If they would have a uniform strategy-selection method however, common knowledge of a winning strategy would be a sufficient requirement. For a single agent things are easier; it is already in a winning position if it knows a winning feasible strategy.

The original paper on ATEL speaks about the paradigm of 'planning as model-checking': strategies for an agent can be obtained by model-checking a goal property. This is an attractive idea, since the authors also show that the model-checking problem for ATEL is tractable. In the same paper, the suggestion was made that knowledge based programs could be generated by model-checking knowledge preconditions. We questioned the plausibility of this suggestion. Instead, we showed that model-checking a formula $\langle\!\langle \ldots \rangle\!\rangle^{\mathrm{f}} \varphi$ will generate the desired knowledge based programs.

We also gave a sketch of a general path that may be followed if one wants to take up model-checking for agents with perfect recall or other mental capabilities. The path consists of a first stage where scenarios are modeled as interpreted environments; a second stage where the corresponding concurrent epistemic game structure is automatically generated from the interpreted environment according to the mental capabilities of the agent; and finally the third stage where the actual model-checking is done. Obviously, the model-checking problem is the first suggestion for further research on Feasible ATEL.

Feasible strategies are a natural concept. So is the notion that an agent should know its feasible strategy. If we use ATEL to reason about real-life situations, the question of whether an agent knows his winning feasible strategy will often be crucial. Therefore the concepts defined in Feasible ATEL

are a natural and powerful addition to ATEL.

List of Figures

3.1 The game structure for KQA-1 23
3.2 The game structure with epistemic relations for KQA-1 . . . 24

4.1 The game that led to some confusion. 34
4.2 The game with reflexive epistemic relations. 36
4.3 A finite interpreted environment and its corresponding game tree. 38
4.4 A game tree for a 'plain' agent (reflexive relations not drawn). 40
4.5 A game tree for an agent with perfect recall (reflexive relations not drawn). 40
4.6 A game tree for a 'synchronous' agent (reflexive relations not drawn). 41
4.7 Part of the unraveled game structure for KQA-1b 44
4.8 Part of the unraveled game structure for KQA-1. 45
4.9 The game structure for the game KQA-1c. 47
4.10 The transitions of KQA-2. 48
4.11 The epistemic accessibility relations of KQA-2. 48
4.12 The game structure for KQA-2b. 52
4.13 The game structure for "Matching Pennies". 56

List of lemmas

1 $KQA1 \models \langle\!\langle Anna \rangle\!\rangle^{\mathrm{f}} \bigcirc win_{Anna}$ 32

2 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C} \Psi \quad \Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi$ 51

3 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi \quad \Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a} \Psi$ 51

4 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C} \Psi \quad \Rightarrow \quad S, q \models C_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C} \Psi$ 52

5 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi \quad \not\Rightarrow \quad S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi$ 53

6 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi \quad \Rightarrow \quad S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi$ 53

7 $S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a} \Psi \quad \Rightarrow \quad S, q \models K_a \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a} \Psi$ 53

8 $S, q \models C_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C} \Psi$ 54

8 $S, q \models E_\Gamma \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{E} \Psi$ 54

8 $S, q \models K_a \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}} \Psi \quad \not\Rightarrow \quad S, q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{K_a} \Psi$ 54

9 If Γ has a uniform strategy-selection method and $q \models \langle\!\langle \Gamma \rangle\!\rangle^{\mathrm{f}}_{C} \varphi \quad \Rightarrow$ Γ is in a winning position in state q 57

10 $q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{K_a} \varphi \iff q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{E} \varphi \iff q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{C} \varphi$ 58

11 $q \models \langle\!\langle a \rangle\!\rangle^{\mathrm{f}}_{K_a} \varphi \Rightarrow$ agent a is in a winning position in state q . 58

Bibliography

[AHK97] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100–109, Florida, October 1997.

[AHM+98] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. Mocha: Modularity in model checking. In *CAV 1998: Tenth International Conference on Computer-aided Verification, (LNCS Volume 1427)*, pages 521–525. Springer-Verlag: Berlin, Germany, 1998.

[Bar46] R. C. Barcan. A functional calculus of first order based on strict implication. *jsl*, 11:1–16, 1946.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Logics of Programs — Proceedings 1981 (LNCS Volume 131)*, pages 52–71. Springer-Verlag: Berlin, Germany, 1981.

[CES83] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. In *Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages*, pages 117–126. ACM, ACM, January 1983.

[DAFM03] D.Gabbay, A.Kurucz, F.Wolter, and M.Zakharyaschev. Many-Dimensional Modal Logics: Theory And Applications. Unpublished book, 2003.

[Duc01] Ho Ngoc Duc. *Resource-Bounded Reasoning about Knowledge.* PhD thesis, Faculty of Mathematics and Informatics, University of Leipzig, April 2001.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[FG97] S. Franklin and A. Graesser. Is it an agent, or just a program? In J. P. Müller, M. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III (LNAI Volume 1193)*, pages 21–36. Springer-Verlag: Berlin, Germany, 1997.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* The MIT Press: Cambridge, MA, 1995.

[Fis96] M. Fisher. An introduction to executable temporal logic. *The Knowledge Engineering Review*, 11(1):43–56, 1996.

[GT99] F. Giunchiglia and P. Traverso. Planning as model checking. In S. Biundo and M. Fox, editors, *Recent Advances in AI Planning (LNAI Volume 1809)*, pages 1–20. Springer-Verlag: Berlin, Germany, 1999.

[Hin62] J. Hintikka. *Knowledge and Belief.* Cornell University Press: Ithaca, NY, 1962.

[HM84] J. Y. Halpern and Y. O. Moses. Knowledge and common knowledge in distributed environments. In *Proceedings of the 3rd ACM Conference on Principles of Distributed Computing.* ACM Press, 1984.

[Jam03] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Keplicz and R. Verbrugge, editors, *Proceedings of the workshop on Formal Approaches to Multi-Agent Systems*, pages 133–140, April 2003.

[JvdH03] W. Jamroga and W. van der Hoek. Agents that know how to play, 2003. Manusript, submitted.

[Kri63] S. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, Austin, TX, 1984.

[Lew18] Clarence Irving Lewis. *A Survey of Symbolic Logic*. Univ. of California Press, Berkeley, Berkeley, 1918. Reprint of Chapters I–IV by Dover Publications, 1960, New York.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag: Berlin, Germany, 1992.

[MvdH95] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press: Cambridge, England, 1995.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the Eighteenth IEEE Symposium on the Foundations of Computer Science*, pages 46–57, 1977.

[PR89] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *Proceedings of the Sixteenth International Colloquium on Automata, Languages, and Programs*, 1989.

[Pri57] Arthur N. Prior. *Time and modality*. Oxford University Press, Oxford, UK, 1957.

[Pri67] Arthur N. Prior. *Past, Present and Future*. Oxford University Press, Oxford, 1967.

[Pri68] Arthur N. Prior. *Papers on Time and Tense*. Oxford University Press, Oxford, 1968.

[Sho88] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. The MIT Press: Cambridge, MA, 1988.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice-hall International, Inc., 1996.

[Var01] M. Y. Vardi. Branching vs. linear time: Final showdown. In T. Margaria and W. Yi, editors, *Proceedings of the 2001 Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001 (LNCS Volume 2031)*, pages 1–22. Springer-Verlag: Berlin, Germany, April 2001.

[vdHW02] Wiebe van der Hoek and Michael Wooldridge. Tractable multiagent planning for epistemic goals. In Maria Gini, Toru Ishida, Cristiano Castelfranchi, and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1167–1174. ACM Press, July 2002.

[vdMS99] R. van der Meyden and N. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science (LNCS Volume 1738)*, pages 432–445. Springer-Verlag: Berlin, Germany, 1999.

[Vis98] Willem C. Visser. *Efficient CTL* Model Checking Using Games And Automata*. PhD thesis, UMCS, june 1998.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *lics86*, pages 332–344, 1986.

[WJ95] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.