Game-Theoretic and Computational Aspects of Concurrent Game Models

Steen Vester

Kongens Lyngby 2013
IMM-M.Sc.-2013-11

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2013-11

Summary (English)

Concurrent games provide a very expressive way of modelling the interaction between agents in reactive systems with infinite behavior. We analyze the model-checking problem of the alternating-time temporal logics ATL and ATL^* interpreted over concurrent game models. This problem has applications in program synthesis and formal verification of concurrent programs, distributed systems and communication protocols. In alternating-time temporal logic the semantics is typically defined such that players can either remember the whole history of a game (perfect recall semantics) or remember nothing about the history of a game (memoryless semantics). Perfect recall strategies have the advantage that many games can be solved when players have access to infinite memory, but this also leads to high complexity (and undecidability in some cases) of the model-checking problem. With memoryless strategies many fewer games can be solved, but on the other hand the complexity of model-checking is much lower, due to the vast restriction of the strategy space. We propose to consider finite-memory semantics as an intermediate case. This is introduced both with a bounded memory size and with an unbounded memory size. This notion is introduced to be able to solve more games than with memoryless semantics, but without getting as high complexity as for perfect recall semantics. Therefore we study the expressiveness of the new types of semantics as well as the complexity of the model-checking problem. This is done for ATL and ATL^* in both complete and incomplete information concurrent games.

Summary (Danish)

Samtidige spil er en meget udtryksfuld måde at modellere interaktionen mellem agenter i reaktive systemer med uendelig opførsel på. Vi analyserer model-checking problemet for de alternerende-tids temporallogikker ATL og ATL^* fortolket over samtidige spilmodeller. Dette problem har anvendelser i program syntese og formel verifikation af samtidige programmer, distribuerede systemer og kommunikationsprotokoller. I alternerende-tids temporal logik er semantikken typisk defineret så spillerne enten kan huske hele historien i et spil (perfekt hukommelsessemantik) eller ikke kan huske noget fra historien overhovedet (hukommelsesløs semantik). Perfekt hukommelsesstrategier har den fordel, at mange spil kan løses, når spillere har adgang til uendelig hukommelse, men de leder samtidig til høj kompleksitet (og ubestemmelighed i nogle tilfælde) af model-checking problemet. Med hukommelsesløse strategier kan mange færre spil løses, men på den anden side er kompleksiteten af model-checking problemet meget lavere på grund af den kraftige reduktion af strategirummet. Vi foreslår at betragte endelig hukommelsessemantik som et mellemliggende tilfælde. Det introduceres både med en begrænset hukommelsesstørrelse og med en ubegrænset hukommelsesstørrelse. Dette begreb introduceres for at kunne løse flere spil end med hukommelsesløs semantik, men uden at få så høj kompleksitet som for perfekt hukommelsessemantik. Derfor studerer vi udtryksmuligheder med de nye typer semantikker samt kompleksiteten af model-checking problemet. Dette gøres for ATL og ATL^* i samtidige spil med både fuldstændig og ufuldstændig information.

Preface

This thesis on *Game-Theoretic and Computational Aspects of Concurrent Game Models* was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Informatics. The thesis was written with Valentin Goranko as supervisor.

The thesis deals with game-theoretic and computational aspects of concurrent game models and in particular focuses on the model-checking problem of alternating-time temporal logic and memory requirements for sure-winning strategies in concurrent games

The thesis consists of an introduction to concurrent game structures and alternating-time temporal logic. In addition, two new types of semantics for this logic are introduced in which strategies are restricted to only using a finite amount of memory. In this context, the model-checking problem is analyzed and the complexity is compared to existing results on traditional semantics.

Lyngby, 01-March-2013

Steen Vester

Acknowledgements

I would like to thank my supervisor Valentin Goranko from IMM, DTU for the support and interesting discussions throughout the preparation of this thesis as well as during my stay at ENS Cachan in 2011-2012. I would also like to thank my former supervisors Patricia-Bouyer Decitre and Nicolas Markey from Laboratoire Spécification et Vérification at ENS Cachan who really woke my interest in infinite games and gave me great support during a research internship from March 2012 until October 2012. I would also like to thank my former tutor Jørgen Villadsen from IMM, DTU who supervised the preparation of my Bachelor thesis and has been very supportive during my studies the previous years. I would also like to thank Mikko Berggren Ettienne, Pernille Saugmann, my father Jens Vester and my girlfriend Pil Maria Saugmann for helpful comments on the thesis. Finally, I would like to thank my family and girlfriend for always giving great support.

Contents

Chapter 1

Introduction

In the past 15 years concurrent game structures have gained an increasing interest as a tool for modelling reactive systems in which multiple agents interact, see e.g. [AHK02, dAH00, dAHM01, Sch04]. Concurrent games generalize Kripke structures which have been used in computer science for a long time to model the behavior of concurrent programs and distributed systems. In particular *model-checking* of logical languages such as the linear-time temporal logic LTL and the branching-time temporal logics CTL and CTL^* over Kripke structures has been very popular for the formal verification of temporal properties of computing systems since the late 1970's, see e.g. [Pnu77, EH86, CES86, CE81, CGP01, BK08]. This technique has been used, mainly, as a verification tool to check properties of [CGP01, BK08]

- Systems with multiple threads running concurrently with non-deterministic interleaving
- Distributed systems and communication protocols
- Hardware design.

All these applications require that a system has already been designed and/or implemented and needs to be checked for errors. Instead of modelling the behavior of a fixed program, we use concurrent game structures to model all the

possible behaviors of multiple devices in an environment of interaction. Kripke structures are extended in this way because we are interested in doing *program synthesis*, which is the discipline of automatically generating a program from a given formal specification. And in order to generate such a program for some device(s), it must be specified what the capabilities of the device(s) are and how they affect the state of the world. This is the purpose of the concurrent game structure.

Next, we introduce the alternating-time temporal logics ATL and ATL^* [AHK97, AHK02] which generalize CTL and CTL^* respectively (ATL^* also generalizes LTL). These logics are used to specify the behavior that we want programs to exhibit. We will see later that model-checking formulae of alternating-time temporal logic over concurrent game structures consists of asking whether there exists strategies for coalitions of players in a game that make sure a given property is satisfied. The strategies of the players in the game correspond directly to programs of the devices that are modelled by the players. The model-checking problem then corresponds to asking whether there exists a program with some desired behavior. Since we are also interested in actually obtaining a program with the desired behavior and not just to know whether there exists one, it is important that the algorithms we find are constructive and produce actual witness strategies with the desired behavior. All the model-checking algorithms we consider in this report are constructive in this sense.

Alternating-time temporal logic has traditionally been used with two different types of semantics. One where players can always remember the whole history of the play and one where players are not allowed to use any memory at all. These semantics are known as perfect recall semantics and memoryless semantics respectively [AHK02, Sch04, JD08, Jam08]. This is a tradeoff between the complexity of the model-checking problem and the number of games where winning strategies can be found. Without any memory, fewer games can be solved, but on the other hand the model-checking is far more efficient than with perfect recall.

The main objective of this thesis is to introduce finite-memory semantics as an intermediate case, where players are allowed to store and update a finite amount of memory about the history of the game during play. We distinguish between two such types of semantics, one where the memory size is bounded, and one where the memory size is finite, but unbounded. These will be called bounded-memory semantics and finite-memory semantics respectively. We wish to investigate the model-checking complexity as well as expressiveness of these new types of semantics compared to perfect recall and memoryless semantics. These problems will be analyzed for both ATL and ATL^* and complete and incomplete information concurrent games.

The motivation for analyzing capabilities and complexity of strategies with finite memory is that the complexity of model-checking with perfect recall semantics is quite high in most cases. Model-checking both ATL and ATL^* with perfect recall semantics is undecidable for concurrent games with incomplete information [AHK02, DT11] and $2EXPTIME$-complete for ATL^* with complete information [AHK02] which makes it infeasible to use in practice. For memoryless strategies however, the model-checking is in Δ_2^p-complete for ATL with incomplete information and $PSPACE$-complete for ATL^* with both complete and incomplete information. While this might not seem very good either, it is closely related to the $PSPACE$-completeness of LTL model-checking, where algorithms have been developed that work very well in practice and have verified systems with quite large state spaces [CGP01, BK08]. We wish to investigate if we can avoid a large increase in complexity by using strategies with finite memory instead of memoryless strategies. Since the technique is designed for program synthesis of distributed systems, it actually seems to be a better model to only allow strategies with access to finite memory. This is because the devices that we wish to program do not have access to an infinite amount of memory. Thus, it does not seem to be an unreasonable restriction of the strategy space to only consider strategies with finite memory.

In chapter 2 we introduce concurrent game structures, and in chapter 3 we introduce alternating-time temporal logic, including the different types of semantics. In chapter 4 we analyze the difference in expressiveness between the different types of semantics, and in chapter 5 we present algorithms and complexity results for the model-checking problems.

Chapter 2

Concurrent Games

2.1 Concurrent game structures

We start by introducing *concurrent game structures* which have been analyzed by a number of other authors, including [AHK02, dAH00, dAHM01, Sch04]. Concurrent game structures are used as models of interaction between a finite number of players (agents) with infinite duration. They generalize notions like turn-based games and Kripke models which are used for modeling, verification and synthesis of concurrent programs and distributed systems. They also generalize the classical game-theoretic notions of finite strategic games and finite extensive-form games. A concurrent game is played on a finite graph by a finite number of players, where the nodes represent the different states of the game and edges represent transitions between game states. An example of a concurrent game $\mathcal{G}_1$ can be seen in Figure 2.1 which will be used as a running example. The game is played an infinite number of rounds and in each round the players must concurrently and independently choose an action that is avilable. The choices of actions of all the players then determine the successor of the current game state. In the figure, each edge is labelled with the tuple of actions that triggers its corresponding transition. For instance, the play goes from state s_2 to state s_3 if player 1 chooses $-$ and player 2 chooses $*$. In $\mathcal{G}_1$ the play starts in state s_0, whic is marked by the token $\bullet$. In the first round player 2 can choose $+$ or $-$ which decides whether the play continues to s_1 or s_2. Player 1 can only

choose the action $*$. After the first transition, player 1 chooses the successor state in every round, since player 2 only has one legal action in all other states than s_0.

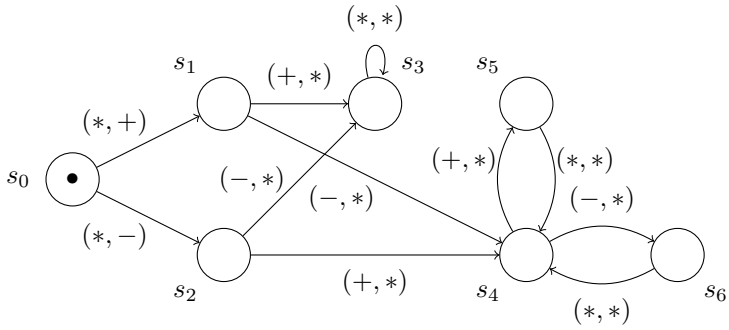

Figure 2.1: Concurrent game structure $\mathcal{G}_1$

Formally, we define a concurrent game structure as follows

Definition 1 *A concurrent game structure (CGS) with n players*

$$\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$$

consists of

- States - *A finite non-empty set of states.*
- $\text{Agt} = \{1, ..., n\}$ - *A finite non-empty set of players.*
- Act - *A finite non-empty set of actions.*
- $\text{Mov} : \text{States} \times \text{Agt} \to 2^{\text{Act}} \setminus \{\emptyset\}$ - *A function specifying the legal actions at a given state of a given player.*
- $\text{Tab} : \text{States} \times \text{Act}^n \to \text{States}$ - *A transition function defined for each $(a_1, ..., a_n) \in \text{Act}^n$ and state s such that $a_j \in \text{Mov}(s, j)$ for $1 \leq j \leq n$.*

Unless otherwise noted, we will implicitly assume from now on that the players in a game are named $1, ..., n$ where $n = |\text{Agt}|$. Note that every play must have at least one legal action in each state. The transition function Tab is defined for each state and all legal tuples of actions in that state. We also refer to such legal tuples of actions as *moves*.

Compared to turn-based games, the important addition is that players must choose actions at the same time independently to decide the behavior of the game. Compared to Kripke frames the important addition is that several players (possibly with different objectives) interact to decide the behavior of an execution rather than just a single player. These generalizations provide us with more expressiveness, but also come with an increase in computational complexity in some cases.

To add meaning to concurrent game structures we introduce the concept of a *concurrent game model* which consists of a concurrent game structure as well as a labeling of the states in the structure with propositions from some finite set Prop of proposition symbols. The idea behind introducing propositions is to be able to express properties of the play of a game as well as objectives of the players using logics over the propositions.

Definition 2 *A concurrent game model (CGM) is a pair* $(\mathcal{G}, \pi)$ *where* $\mathcal{G}$ *is a concurrent game structure and* $\pi : \text{States} \to \mathcal{P}(\text{Prop})$ *is a labeling function.*

An example of a CGM can be seen in Figure 2.2.

2.1.1 Examples

A CGS and a CGM

In the framework introduced above, the CGS $\mathcal{G}_1 = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$ in Figure 2.1 is defined by

- $\text{States} = \{s_0, s_1, ..., s_6\}$
- $\text{Agt} = \{1, 2\}$
- $\text{Act} = \{*, +, -\}$
- $\text{Mov}(s_j, 1) = \{*\}$ when $j = 0$ and $\text{Mov}(s_j, 1) = \{+, -\}$ for $j = 1, ..., 6$.
- $\text{Mov}(s_j, 2) = \{+, -\}$ when $j = 0$ and $\text{Mov}(s_j, 2) = \{*\}$ for $j = 1, ..., 6$.

The transitions of the game can be deduced from the labelled edges between the nodes. For instance $\text{Tab}(s_0, (*, -)) = s_2$, $\text{Tab}(s_3, (*, *)) = s_3$ and $\text{Tab}(s_5, (*, *)) = s_4$.

In Figure 2.2 we illustrate the CGM $\mathcal{M}_1 = (\mathcal{G}_1, \pi_1)$ which consists of the CGS $\mathcal{G}_1$ as already defined and the function π_1 labeling the states in $\mathcal{G}_1$ by proposition symbols from the set Prop $= \{p, q, r\}$. π_1 is defined so $\pi_1(s_4) = \{p\}$, $\pi_1(s_5) = \{r\}$ $\pi_1(s_6) = \{p, q\}$ and $\pi_j = \emptyset$ for $j = 0, 1, 2, 3$.

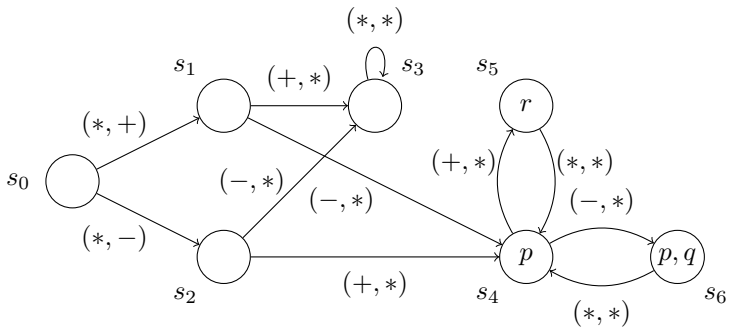

Figure 2.2: Concurrent game model $\mathcal{M}_1 = (\mathcal{G}_1, \pi_1)$

Kripke frames are concurrent game structures

A Kripke frame $\langle W, R\rangle$ where W is the set of worlds and $R = \{(x_1, y_1), ..., (x_m, y_m)\}$ is the binary accessibility relation on W can be modelled as a 1-player concurrent game $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$ where States $= W$, Agt $= \{1\}$. We introduce one action a_j for each pair (x_j, y_j) in R, i.e. Act $= \{a_1, ..., a_m\}$. For each pair $(x_j, y_j) \in R$ we let a_j be the action that takes the play from x_j to y_j, i.e. $\text{Mov}(s, 1) = \{a_j | (x_j, y_j) \in R \wedge x_j = s\}$ for all $s \in W$ and $\text{Tab}(x_j, a_j) = y_j$ for $1 \leq j \leq m$. Kripke structures can thus be modelled as 1-player game structures where the single player controls the play.

When model-checking logics such as the linear-time temporal logic LTL [Pnu77] or the computation tree logics $CTL/CTL*$ [CE81, EH86] in Kripke models, the Kripke models are often used to model the possible behaviors of a program (or several concurrently running programs). Then the worlds correspond to system states and the transitions correspond to possible choices of the environment or a scheduler which are regarded as non-deterministic. The purpose is then to decide whether such a model of a program satisfies a specification given in LTL or CTL/CTL^*. In particular, one usually wants to know whether *all* executions of the program satisfiy the specification or whether there *exists* some execution that does not satisfy the specification. This can also be done with concurrent game structures since they generalize Kripke structures. However, there are also other possibilities. For instance, we can use a concurrent game structure

to model both the possible choices of the environment as well as the choices of some device we wish to program. In this case we might be interested in whether there exists a strategy for the device player to make sure that the play satisfies some specification no matter what the environment player does. Such a strategy is interesting to search for, since it corresponds to a program of the device that matches the specification no matter how the environment reacts. In other words, an algorithm to find such a strategy is an automatic way to program a device to match a given specification. This is a particular application of concurrent games in *program synthesis*. The same idea can also be used to motivate the use of concurrent games in the synthesis of more complex distributed systems where different devices have different objectives to satisfy.

Turn-based games are concurrent game structures

Turn-based games is the fragment of concurrent games where the successor state from any given state during the play is chosen by (at most) a single player. More formally, it is the fragment of concurrent games where for all $s \in \text{States}$ we have $|\text{Mov}(s, j)| > 1$ for at most one player j. Turn-based games have been studied quite extensively in the litterature, e.g. in [GS51, BL69, Mar75, Rei84]. They are of importance in this thesis since we will use results from turn-based games in proofs and algorithms for concurrent games.

2.2 Outcomes and histories

A concurrent game is played an infinite number of rounds. In each round all players simultaneously and independently choose a legal action, where the legal actions are specified by the Mov function. The move chosen by the players uniquely determines the successor state which is specified by the Tab function. This infinite sequence of moves produces an infinite sequence of states called a *play*, a *run* or an *outcome* of a game. More formally, the set of outcomes $\text{Out}_{\mathcal{G}}(s)$ of a CGS $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$ from state s is defined as

$$\text{Out}_{\mathcal{G}}(s) = \{s_0 s_1 ... \in \text{States}^{\omega} | s_0 = s \text{ and } \forall j \geq 0. \exists m \in \prod_{k=1}^{|Agt|} \text{Mov}(s_j, k). \text{Tab}(s_j, m) = s_{j+1}\}$$

We usually omit the subscript $\mathcal{G}$ when the game is clear from the context. The set of all outcomes is defined as $\text{Out}_{\mathcal{G}} = \bigcup_{s' \in \text{States}} \text{Out}_{\mathcal{G}}(s')$. Next, we define

the set of histories $\text{Hist}_{\mathcal{G}}(s)$ starting in s which consists of all finite sequences of states that is the prefix of some outcome starting in s, i.e.

$$\text{Hist}_{\mathcal{G}}(s) = \{s_0 s_1 ... s_k | s_0 = s \text{ and } \exists \rho \in \text{Out}_{\mathcal{G}}(s).s_0 s_1 ... s_k \leq \rho\}$$

where $\leq$ is a prefix operator. As for outcomes we define the set of all histories as $\text{Hist}_{\mathcal{G}} = \bigcup_{s' \in \text{States}} \text{Hist}_{\mathcal{G}}(s')$. The reason why outcomes are important is that the objectives of a player is typically given as a subset of the set of outcomes of a game. Histories, on the other hand, are important since a player must decide which action to choose depending of the finite history of the game in any given round. A *path* is either a history or an outcome of a game.

For sequences $\rho = \rho_0 \rho_1 ...$ of states we denote by ρ_j the $(j+1)$th state in ρ. We denote by $\rho_{\leq k}$ the prefix $\rho_0 \rho_1 ... \rho_k$ of ρ. When ρ is a finite sequence, i.e. $\rho = \rho_0 \rho_1 ... \rho_m$ for some m, we define the length $|\rho|$ of ρ as the number of transitions in the path, so in this case $|\rho| = m$. $\text{last}(\rho) = \rho_{|\rho|}$ is used to denote the last state in the finite sequence ρ. We define the set $\text{Occ}(\rho) = \{s \in \text{States} | \exists j \geq 0.\rho_j = s\}$ as the set of states occuring on the path ρ and the set $\text{Inf}(\rho) = \{s \in \text{States} | \forall j \geq 0. \exists k > j.\rho_k = s\}$ as the set of states that occur infinitely often on the path ρ.

2.3 Strategies

In this section we define strategies for players in a concurrent game. In addition, we focus on subclasses of strategies in which the players are not able to remember everything that happened earlier in the game. All strategies considered in this thesis are pure strategies, which are strategies such that players always choose one specific action, given a particular history. This is in contrast to probabilistic strategic which are often used in game theory. The reason for this choice is our focus on sure-winning strategies where probabilistic strategies are not needed.

2.3.1 Strategies with perfect recall

A strategy of a player j in a CGS $\mathcal{G}$ is a mapping $\sigma_j : \text{Hist}_{\mathcal{G}} \to \text{Act}$ such that for any history $\rho \in \text{Hist}_{\mathcal{G}}$ it holds that $\sigma_j(\rho) \in \text{Mov}(\text{last}(\rho), j)$. Thus, a strategy σ assigns to any given history h a legal action in the last state of the history, which is the choice of action of a player utilizing strategy σ after observing the history h. A strategy for a coalition A of players is simply a tuple $(\sigma_a)_{a \in A}$ of

strategies, one for each player. The set of all strategies for player j is called Strat_j^R. The symbol R is used in the litterature on alternating-time temporal logic (ATL) and will be explained in the next chapter. The set of strategies will also be referred to as strategies with perfect recall, since a player can use the complete history of a game until the decision point to base his decision on. Given strategies $\sigma_A = (\sigma_a)_{a \in A}$ for a subset A of agents we define $\text{Out}_{\mathcal{G}}(s, \sigma_A)$ as the set of outcomes that are possible, when the players in A play according to σ_A. Formally,

$$\text{Out}_{\mathcal{G}}(s, \sigma_A) = \{\rho \in \text{Out}_{\mathcal{G}}(s) | \forall j \geq 0. \exists m \in \prod_{k=1}^{|\text{Agt}|} \text{Mov}(\rho_j, k). \\ \text{Tab}(\rho_j, m) = \rho_{j+1} \text{ and } \forall a \in A. m_a = \sigma_a(\rho_{\leq j})\}$$

We define $\text{Out}_{\mathcal{G}}(\sigma_A) = \bigcup_{s \in \text{States}} \text{Out}_{\mathcal{G}}(s, \sigma_A)$ as all possible outcomes given that players in A play according to σ_A. When a strategy is specified for all players, the outcome from a given state is a singleton set.

We note that there are uncountably many perfect recall strategies. One can see this by considering a game in which player 1 can choose one of the actions $0, 1, ..., 9$ in every step of an infinite game. There is an injective mapping from the real numbers in the interval $[0, 1]$ to the set of strategies for player 1, namely the mapping sending the real number

$d_1 \cdot 10^{-1} + d_2 \cdot 10^{-2} ...$

where $d_j \in \{0, ..., 9\}$ for all $j \geq 1$ to the strategy σ such that $\sigma(\rho) = d_{|\rho|+1}$ for all histories ρ.

2.3.2 Memoryless strategies

The memoryless (also called positional or state-based) strategies constitute an important subset of the set of strategies. A memoryless strategy is a strategy in which a player only chooses his action based on the last state of play, which means that he does not use any memory about the history of play to decide on his choice of action. More formally, a strategy σ is memoryless if $\sigma(\rho) = \sigma(\rho')$ for all histories $\rho, \rho' \in \text{Hist}_{\mathcal{G}}$ with $\text{last}(\rho) = \text{last}(\rho')$. We denote the set of memoryless strategies for player j Strat_j^r.

This class of strategies is important, since many types of games can be shown to have winning strategies if and only if they have memoryless winning strate-

gies. This gives two main advantages. First, it means that the strategies are easy to understand and can be implemented on a machine without the need of memory to track the history. Second, it reduces the search space of strategies vastly since there are at most $|\text{Act}|^{|\text{States}|}$ different memoryless strategies for a player, whereas there are uncountably many perfect recall strategies. This reduction of the search space naturally leads to algorithms of lower complexity when searching for winning strategies. We will see examples of this in later chapters.

2.3.3 Finite-Memory Strategies

As an intermediate case between memoryless and perfect recall strategies we introduce finite-memory strategies. In finite-memory strategies a finite amount of memory about the history of the game is allowed to be saved in order to choose actions later in the game. The finite memory can be updated each time there is a transition in the game. The reason to introduce finite-memory strategies is that memoryless strategies are in many cases not sufficient for winning. In addition, finite-memory strategies can be implemented on a machine. In this thesis we focus on the difference in computational complexity between the different classes of strategies as well as look at classes of games in which different amounts of memory are needed for winning strategies. We use Mealy machines (or Mealy automata) [Mea55] as models of finite-memory strategies as is done in [BL69, PR89, PR90, Tho95, DJW97, Zie98][1]. Mealy machines are deterministic finite automata with output on the transitions. Formally they are defined as follows

Definition 3 *A Mealy machine (Mealy automaton) A is a 6-tuple*

$$A = (M, m_0, \Sigma, \Gamma, T, G)$$

where

- M *is a finite, non-empty set of states*
- $m_0 \in S$ *is the initial state*
- Σ *is a finite, non-empty set of input symbols*
- Γ *is a finite, non-empty set of output symbols*
- $T : S \times \Sigma \to S$ *is a transition function*

[1] This way of representing finite memory is not always called a Mealy machine, but the techniques used in the references are essentially the same

- $G : S \times \Sigma \to \Gamma$ *is an output function*

We require that the functions T and G are total.

In our framework a Mealy machine $A = (M, m_0, \Sigma, \Gamma, T, G)$ represents a finite-memory strategy. The set of states M corresponds to possible values of the memory for a strategy at a given point. We also call these memory states. The initial state m_0 is then the initial memory value of the strategy. The input symbols are states from the CGS $\mathcal{G}$ on which the game is played and the output symbols are actions in $\mathcal{G}$. The transition function T takes the current memory and the state that was last observed in the game as input and updates the memory by a transition to a new memory value. The output function G takes the same input as T and outputs the action chosen by the strategy represented by the Mealy machine. More precisely, we say that a strategy σ_j for player j is a *finite memory strategy* in $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$ if there exists a Mealy machine $A = (M, m_0, \text{States}, \text{Act}, T, G)$ such that

$$\sigma_j(\pi \cdot p) = G(\mathcal{T}(m_0, \pi), p)$$

for all $\pi \in \text{Hist}_{\mathcal{G}}$ and all $p \in \text{States}$ where $\mathcal{T}$ is defined recursively by $\mathcal{T}(s, \rho) = T(s, \rho_0)$ for any state s and any ρ with $|\rho| = 0$ and $\mathcal{T}(s, \rho) = T(\mathcal{T}(s, \rho_{\leq |\rho|-1}), \text{last}(\rho))$ for any s and any path ρ with $|\rho| \geq 1$. Intuitively $\mathcal{T}$ is the function that repeatedly apply the transition function T on a sequence of inputs to calculate the memory state after a given history. We call $\mathcal{T}$ the repeated transition function. We say that σ_j is a k-memory strategy if the number of states of the Mealy machine is k. We also say that the strategy σ_j is represented by the Mealy machine A. We denote the set of finite-memory strategies for player j Strat_j^F and the set of k-memory strategies for player j $\text{Strat}_j^{F_k}$. Thus, $\text{Strat}_j^F = \bigcup_{k \geq 1} \text{Strat}_j^{F_k}$. In addition, we have that the memoryless strategies are exactly the finite-memory strategies with one memory state, i.e. $\text{Strat}_j^{F_1} = \text{Strat}_j^r$.

The number of different k-memory strategies is finite and bounded by

$$((k^{|\text{States}|} \cdot |\text{Act}|^{|\text{States}|})^k) = (k \cdot |\text{Act}|)^{k \cdot |\text{States}|}$$

which means that the number of k-memory strategies is finite for a fixed k just as for memoryless strategies. This means that the set of finite-memory strategies is infinite, but countable.

2.3.4 Examples

Consider again the CGM $\mathcal{M}_1 = (\mathcal{G}_1, \pi_1)$ in Figure 2.2. For player 2 we define a memoryless strategy σ_2 where he chooses $+$ in the initial state s_0 and $*$ in all others states. A Mealy automaton A_2 representing σ_2 is shown in Figure 2.3.

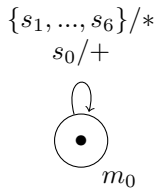

Figure 2.3: Mealy automaton A_2 representing the memoryless strategy σ_2

In the graphical representation of Mealy automata the nodes correspond to memory states. There is an edge from a memory-state m_1 to memory-state m_2 labeled with s/a if the strategy with current memory m_1 when seeing the current state in the game s chooses action a and change the memory to m_2. Formally, it means that $T(m_1, s) = m_2$ and $G(m_1, s) = a$ where T and G are the transition function and output function respectively. For instance, this is what we mean by $s_0/+$ in the Figure. For simplicity, we can also write S/a on an edge for a set S of input symbols, given that all the input symbols in S produce the same output and the same transition from the given memory state. This is done with the edge labelled by $\{s_1, ..., s_6\}/*$. We can also label one edge with several labels, when the labels correspond to the same memory update. Finally, the initial memory state is marked by the token $\bullet$.

In Figure 2.4 is the Mealy automaton A_1 representing a strategy σ_1 for player 1 in $\mathcal{M}_1$ where he always chooses $*$ in s_0, s_3, s_5 and s_6, chooses $-$ in s_1 and chooses $+$ in s_2. In addition he chooses $+$ in s_4 when s_4 has occured an odd number of times in the history and chooses $-$ when s_4 has occured an even number of times in the history.

This strategy cannot be represented by a Mealy automaton with a single memory state, since in such a strategy the choice of action when observing state s_4 would always be the same. Thus, σ_1 is not a memoryless strategy. But since it is represented by A_1, it is a finite-memory strategy. In fact, player 1 does not have a memoryless strategy to make sure that both r and q are true at some point during the play, because in a memoryless strategy he will always choose the same action in s_4, no matter what the history of the game is. By using the finite-memory strategy σ_1 though, he can make sure this is actually the case,

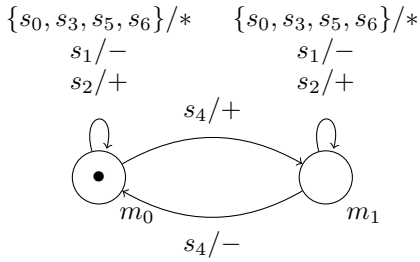

Figure 2.4: Mealy automaton A_1 representing the finite-memory strategy σ_1

no matter which strategy player 2 uses.

2.4 Concurrent game structures with incomplete information

In many situations of interaction between agents, it is not the case that every agent has complete information about the state of the world at all times. Consider for instance a device which has the job of controlling a power plant. The device will receive information about the state of the plant from a number of different sensors. This information gives a good indication of the internal state of the plant, but does not give complete information about what is going on. We would like to model situations like this using the framework of concurrent games, where one player would represent the controller and the other player would represent the environment. Here, the controller must be able to meet certain specifications, even in the absence of complete information about the plant. To do this we propose to enrich the concurrent game structures with equivalence relations $\sim_j$ on states of the game for every player j. The intuition is that for two states s, s' we have $s \sim_j s'$ iff player j cannot distinguish between the two states. This model has also been analysed in the litterature (e.g. [Sch04, JD08, ÅW09, BDJ10] under the names concurrent game structure with incomplete information (iCGS) and concurrent epistemic game structure (CEGS). We adopt the former name and define the notion formally as follows

Definition 4 *A concurrent game structure with incomplete information (iCGS) with n players*

$$\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab}, (\sim_j)_{1 \leq j \leq n})$$

consists of

- States - *A finite non-empty set of states.*
- $\text{Agt} = \{1, ..., n\}$ - *A finite non-empty set of players.*
- Act - *A finite non-empty set of actions.*
- $\text{Mov} : \text{States} \times \text{Agt} \to 2^{\text{Act}} \setminus \{\emptyset\}$ - *A function specifying the legal actions at a given state of a given player.*
- $\text{Tab} : \text{States} \times \text{Act}^n \to \text{States}$ - *A transition function defined for each* $(a_1, ..., a_n) \in \text{Act}^n$ *and state* s *such that* $a_j \in \text{Mov}(s, j)$ *for* $1 \leq j \leq n$.
- $\sim_j \subseteq \text{States} \times \text{States}$ - *An equivalence relation on the set of states for each player* j. *We require for each player* j *that* $\text{Mov}(s, j) = \text{Mov}(s', j)$ *whenever* $s \sim_j s'$

For each player j, the relation $\sim_j$ induces a set of equivalence classes $[]_j$ of States. We denote by $[s]_j$ the class that state s belongs to for player j. These classes are refered to as the information sets of player j. Since the set of legal actions of player j is required to be the same in states from the same information set, we can define $\text{Mov}([s]_j, j) = \text{Mov}(s, j)$ for all states s.

An iCGS $\mathcal{G}_2$ which is equivalent to $\mathcal{G}_1$ from Figure 2.1 except that s_1 and s_2 are indistinguishable to player 1 can be seen in Figure 2.5. The information sets for a player j which are not singletons are drawn by dashed ellipses around the states they contain and labeled with j. In the figure, one can see that $s_1 \sim_1 s_2$, which means that player 1 cannot distinguish between states s_1 and s_2. We also write $[s_1]_1 = [s_2]_1 = \{s_1, s_2\}$.

Note that the notion of iCGS generalizes that of a concurrent game structure, which is obtained by defining $\sim_j$ for all j such that $s \sim_j s'$ for two states $s, s' \in \text{States}$ if and only if $s = s'$. Most of the concepts from concurrent game structures naturally transfer to concurrent game structures with incomplete information. A concurrent game model with incomplete information (iCGM) over a finite set Prop of proposition symbols is defined as follows.

Definition 5 *A concurrent game model with incomplete information (iCGM) is a pair* $(\mathcal{G}, \pi)$ *where* $\mathcal{G}$ *is an iCGS and* $\pi : \text{States} \to \mathcal{P}(\text{Prop})$ *is a labeling function.*

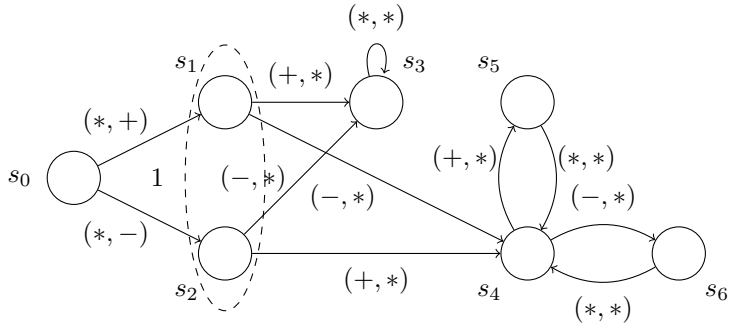

Figure 2.5: Concurrent game structure $\mathcal{G}_2$ with incomplete information

The set of outcomes and the set of histories of an iCGS are defined in the same way as for a CGS. However, the major difference between the two notions is in the strategies available to the players. Since there are states that a given player cannot distinguish between, there are histories which he will not be able to distinguish between as well. For this reason we require that for histories that are indistinguishable to a player, the player must choose the same action in a strategy after these histories because he cannot distinguish between them. To define when histories are indistinguishable, we lift the indistinguishability relation $\sim_j$ for player j to histories such that for two histories ρ and ρ'

$$\rho \sim_j \rho' \text{ if and only if } |\rho| = |\rho'| \text{ and } \rho_k \sim_j \rho'_k \text{ for all } 0 \leq k \leq |\rho|$$

Thus, two histories are indistinguishable to a player if and only if they have the same length and the states on the paths are pairwise indistinguishable. A strategy that always uses the same decision after indistinguishable histories is sometimes called a uniform strategy. However, we take a different, but equivalent, approach where strategies are defined on information histories, which are sequences of information sets corresponding to histories of a play. More formally, the set $\text{Hist}^j_{\mathcal{G}}$ of information histories of player j in iCGS $\mathcal{G}$ are defined as

$$\text{Hist}^j_{\mathcal{G}} = \{[s_0]_j[s_1]_j...[s_k]_j | s_0 s_1 ... s_k \in \text{Hist}_{\mathcal{G}}\}$$

For a (finite or infinite) sequence $\rho = \rho_0\rho_1...$ of states we define the corresponding information sequence for player j by $[\rho]_j = [\rho_0]_j[\rho_1]_j....$

A strategy σ in an iCGS $\mathcal{G}$ for player j is a mapping $\sigma : \text{Hist}^j_{\mathcal{G}} \to \text{Act}$. This definition of a strategy in an iCGM corresponds exactly to the definition of uniform strategies outlined above. The definition of a memoryless strategy is the same as in a CGM with complete information. The definition of a finite-memory strategy is the same except that the input alphabet used for the Mealy automaton representing a strategy for player j is the set of information sets for player j instead of the set of states, since the player now observes information sets and not states.

Consider again the game in Figure 2.5. In this game, unlike the complete information game we studied earlier, player 2 does not have a strategy to make sure that the play goes to s_4. The reason is that in all strategies, player 2 must choose the same action after histories $h_1 = s_0 s_1$ and $h_2 = s_0 s_2$ since $[h_1]_2 = [h_2]_2$. This means that he must choose either $+$ after both histories or $-$ after both histories. No matter which strategy he chooses, player 1 has a strategy that take the play to s_3 instead. This is a typical example of how the amount of information available restricts the set of strategies that are available.

2.5 Winning objectives

In this thesis we focus on sure-winning strategies for coalitions of players for various qualitative objectives. Given an (i)CGS $\mathcal{G}$, an objective $\Omega \subseteq \text{Out}_{\mathcal{G}}$ is a subset of the set of possible outcomes. A strategy σ_A for a coalition A of players is said to be sure-winning (or winning) from state s in $\mathcal{G}$ with objective Ω if $\text{Out}_{\mathcal{G}}(s', \sigma_A) \subseteq \Omega$ for all s' such that $s \sim_j s'$ for some $j \in A$. When such a strategy exists we say that s is winning for coalition A. When σ_A is a strategy for agents in coalition A with objective Ω, we say that the strategy σ_B for $B = \text{Agt} \setminus A$ is a spoiling strategy for σ_A with objective Ω from state s if there exists $j \in A$ and $s' \sim_j s$ such that $\text{Out}_{\mathcal{G}}(s', (\sigma_A, \sigma_B)) \not\subseteq \Omega$. In other words, a spoiling strategy is a counter strategy for the players in $\text{Agt} \setminus A$ such that coalition A using strategy σ_A cannot be sure to win. However, this does not mean that σ_B is sure to win for the complementary objective $\text{Out}_{\mathcal{G}} \setminus \Omega$, only that σ_B works as a counter strategy against the specific strategy σ_A for players in A.

Next, we introduce some classes of objectives that have been analyzed in the litterature. This is done because we apply some of these results later in the thesis. If $\mathcal{M} = (\mathcal{G}, \pi)$ is an (i)CGM we define the following classes of objectives

- *Reachability objectives*, which are given by a set R. The objective is then

defined by $\Omega_{reach}(R) = \{\rho \in \text{Out}_{\mathcal{G}} | R \cap \text{Occ}(\rho) \neq \emptyset\}$, i.e. the set of outcomes where at least one state from R occurs.

- *Safety objectives*, which are given by a set R. The objective is then defined by $\Omega_{safe}(R) = \{\rho \in \text{Out}_{\mathcal{G}} | R \cap \text{Occ}(\rho) = \emptyset\}$, i.e. the set of outcomes where no state from R occurs.

- *Büchi objectives*, which are given by a set R. The objective is then defined by $\Omega_{Buchi}(R) = \{\rho \in \text{Out}_{\mathcal{G}} | R \cap \text{Inf}(\rho) \neq \emptyset\}$, i.e. the set of outcomes where at least one state from R occurs infinitely often.

- *LTL objectives*, which are given by an LTL formula φ. The objective is then defined by $\Omega_{LTL}(\varphi) = \{\rho \in \text{Out}_{\mathcal{G}} | \mathcal{M}, \rho \models_{LTL} \varphi\}$, i.e. the set of outcomes that satisfies the LTL formula φ.

We define a zero-sum 2 player game as a game where player 1 has objective Ω and player 2 has objective $\text{Out}_{\mathcal{G}} \setminus \Omega$. We say that a 2-player zero-sum game is determined for some objective if from every state in the game, there is a player with a sure-winning strategy. Concurrent games are not, in general, determined.

Remark 1. There are many other interesting types of objectives, but with our focus on the alternating-time temporal logic, the types of objectives introduced here are the most important to us. Other interesting qualitative objectives include, but are not limited to, Borel objectives (see e.g. [BL69, Mar75, RCDH07]), ω-regular objectives (see e.g. [Tho96, dAH00, dAHM01, Maz01, RCDH07]), parity objectives (see e.g. [EJ91, Zie98, Maz01] and Müller objectives (see e.g. [Tho95, DJW97, Maz01, GK07]). Quantitative objectives where players seek to optimize utility functions or games with preference relations on outcomes are also used in the litterature to model interesting situations. (see e.g. [Nas50, OR94, NRTV07, SLB09]). These types of objectives have mostly been studied in the context of finite games.

Remark 2. Since concurrent games are not determined, other solution concepts than sure-winning strategies are of great interest, often combined with probabilistic strategies. Examples of this include almost-sure winning strategies, limit-sure winning strategies (see e.g. [dAH00, RCDH07]) as well as equilibrium strategy profiles, such as Nash equilibria (see e.g. [Nas50, OR94, NRTV07, SLB09]) or subgame perfect equilibria (see e.g. [OR94, NRTV07, SLB09]). Equilibrium strategies have mostly been studied in the context of finite games.

Consider the iCGM $\mathcal{M}_2 = (\mathcal{G}_2, \pi_2)$ in Figure 2.6. Let player 1 have the reachability given by the set $\{s_4\}$ and let player 2 have the complementary safety objective given by the same set. This is an example of a game which is not determined, since none of the players have a sure-winning strategy from state

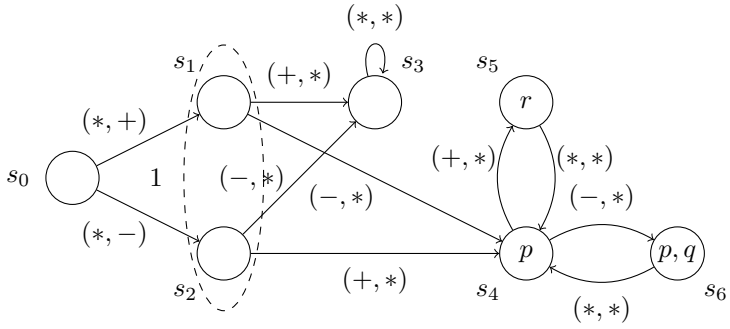

Figure 2.6: iCGM $\mathcal{M}_2 = (\mathcal{G}_2, \pi_2)$

s_0. The reason is that the losing player can change his action in order to win, given that the other player keeps his strategy fixed. An example of a complete information concurrent game that is not determined can be seen in $\mathcal{M}_3$ in Figure 2.7 which is a slight modification of $\mathcal{M}_2$ where the game is not determined from s_0 either. This is due to the fact that players choose actions at the same time. Two-player turn-based games with complete information on other hand are determined for the very general class of Borel objectives [Mar75]. This class includes all ω-regular objectives and therefore all other types of qualitative objectives we have mentioned so far.

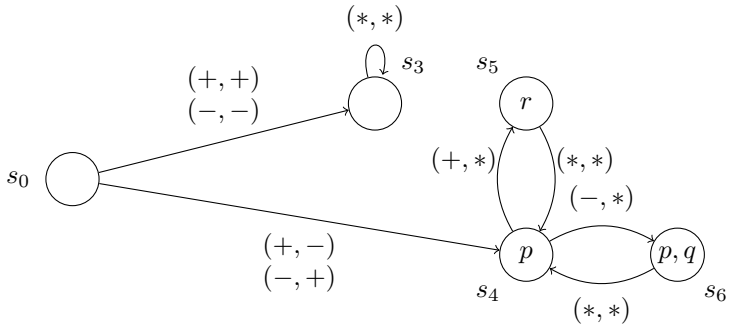

Figure 2.7: CGM $\mathcal{M}_3 = (\mathcal{G}_3, \pi_3)$

Chapter 3

Alternating-time Temporal Logic

3.1 Motivation

In this chapter we introduce the alternating-time temporal logics ATL and ATL^* [AHK97, AHK02] which generalize the computation tree logics CTL [CE81] and CTL^* [EH86] respectively. ATL^* also generalizes the linear-time temporal logic LTL [Pnu77]. Instead of the existence and universal path quantifiers **E** and **A** strategy quantifiers $\langle\langle A \rangle\rangle$ for coalitions A of playeres are introduced. The intuition is that $\langle\langle A \rangle\rangle \varphi$ is true in a state of a concurrent game model if the players in coalition A have a collective strategy σ that makes sure φ is satisfied in the play of the game, no matter what the remaining players of the game do. The strategy quantifiers generalize existential and universal quantifiers since it can be shown that $\mathbf{E} = \langle\langle \mathrm{Agt} \rangle\rangle$ and $\mathbf{A} = \langle\langle \emptyset \rangle\rangle$ where Agt is the set of players in the game structure. The strategy quantifiers give us the possibility of specifying more interesting properties of a system than is possible with computation tree logic. For instance, suppose we have a game describing the possible behavior of processes $\{P_1, ..., P_n\}$ of a system that are all interested in accessing some resource (e.g. database, shared variable) infinitely often. Let these processes correspond to players and let w_i be a proposition that is true when process i has write access. One can specify that in all executions every

process will have write access infinitely often by the CTL^* formula

$$\mathbf{A}(\bigwedge_{j=1}^{n} \mathbf{GF}w_j)$$

However, this condition is quite strong. Let us compare it to the following ATL^* formula (we still use $\mathbf{E}$ and $\mathbf{A}$ for $\langle\langle \mathrm{Agt} \rangle\rangle$ and $\langle\langle \emptyset \rangle\rangle$ respectively)

$$\mathbf{AG}(\bigwedge_{j=1}^{n} \langle\langle P_j \rangle\rangle \mathbf{F}w_j)$$

This formula specifies that in all executions of the system, from any state every process P_j has a strategy to make sure that it will eventually get write access. There might be systems where it does not make sense to enforce that all processes need to have write access infinitely often as described by the CTL^* formula. For instance if the processes don't necessarily all need access infinitely often. But it seems to make sense that all the processes can eventually get access if they need it. Note that both formulae make sure that each process can get write access infinitely often if it wants to, even if the other processes do not act properly. Consider the game between a device D and a (possibly hostile) environment E, where we wish to determine if it is possible to program the device D such that it can make sure some specification φ is always satisfied no matter how E behaves. This is captured by the formula

$$\langle\langle \{D\} \rangle\rangle \mathbf{G}\varphi$$

which is true in a given state s of the game if there is a strategy for D such that φ is always satisfied, no matter how E plays starting in s. Thus, the model-checking of this formula corresponds to deciding whether such a program for D exists. For this reason we are interested in the model-checking problem of ATL^* formulae in concurrent game structures. In the case of a true formula it is also essential that our model-checking algorithm provides a witness strategy, since this will give us an actual program meeting the specification and not only decide whether such a program exists. A witness strategy is a particular strategy with the desired property.

3.2 Syntax

3.2.1 ATL^* Syntax

As in CTL^* the logic ATL^* formulae consists of proposition symbols, logical connectives $(\neg, \vee, \wedge, \rightarrow, \leftrightarrow)$ and temporal operators $(\mathbf{X}, \mathbf{U}, \mathbf{G}, \mathbf{F})$. Unlike CTL^*, there is a strategy quantifier $\langle\langle A \rangle\rangle$ for coalitions A of agents. The informal meanings of the different symbols are as follows

- $\bot$ - The false symbol
- $\neg$ - The negation of a formula
- $\vee, \wedge$ - Disjunction and conjunctions of formulae, respectively.
- $\rightarrow, \leftrightarrow$ - Implication and biimplication of formulae, respectively.
- $\mathbf{X}$ - The "next" operator, where $\mathbf{X}\varphi$ means that φ is true in the next state of a path.
- $\mathbf{U}$ - The "until" operator, where $\varphi_1 \mathbf{U} \varphi_2$ means that φ_1 is true until φ_2 is.
- $\mathbf{F}$ - The "eventuality" operator, where $\mathbf{F}\varphi$ is true if φ is eventually true.
- $\mathbf{G}$ - The "global" operator, where $\mathbf{G}\varphi$ is globally (or always) true.
- $\langle\langle A \rangle\rangle$ - The strategy quantifier, where $\langle\langle A \rangle\rangle \varphi$ is true if coalition A has a strategy to enforce φ.

Since some of the symbols above are inter-definable, we only use a minimal subset of them when specifying the syntax of ATL^*, mostly since it will make case-based proofs shorter. The formulae of ATL^* are constructed from the following grammar

$$\varphi ::= p \mid \bot \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \mathbf{X}\varphi_1 \mid \varphi_1 \mathbf{U} \varphi_2 \mid \langle\langle A \rangle\rangle \varphi_1$$

where $p \in \text{Prop}$ for a fixed, finite set of proposition symbols Prop, $\bot$ is the false symbol, φ_1 and φ_2 are ATL^* formulae and $A \subseteq \text{Agt}$ is a subset of a fixed, finite set Agt of players. As in CTL^* there are state formulae and path formulae. The state formulae are defined as follows

- p is a state formula if $p \in \text{Prop}$.

- $\bot$ is a state formula.
- If φ_1 and φ_2 are state formulae, then $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$ are state formulae.
- If φ_1 is an ATL^* formula and $A \subseteq \mathrm{Agt}$ is a set of players, then $\langle\langle A \rangle\rangle$ is a state formula.

The set of path formulae is the set of all ATL^* formulae constructed from the grammar above. Thus, all state formulae are path formulae, but not all path formulae are state formulae. The symbols not present in the grammar can be defined as follows, where φ_1 and φ_2 are arbitrary ATL^* formulae:

$$
\begin{array}{lcl}
\top & \stackrel{\text{def}}{=} & \neg\bot \\
\varphi_1 \wedge \varphi_2 & \stackrel{\text{def}}{=} & \neg(\neg\varphi_1 \vee \neg\varphi_2) \\
\varphi_1 \rightarrow \varphi_2 & \stackrel{\text{def}}{=} & \neg\varphi_1 \vee \varphi_2 \\
\varphi_1 \leftrightarrow \varphi_2 & \stackrel{\text{def}}{=} & (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1) \\
\mathbf{F}\varphi_1 & \stackrel{\text{def}}{=} & \top \mathrm{U} \varphi_1 \\
\mathbf{G}\varphi_1 & \stackrel{\text{def}}{=} & \neg\mathbf{F}\neg\varphi_1
\end{array}
$$

By using these additional symbols, formulae can often be presented in a more readable way.

3.2.2 ATL Syntax

ATL is the subset of ATL^* that only contains state formulae constructed by the following grammar

$$\varphi ::= p \mid \bot \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \langle\langle A \rangle\rangle \mathbf{X}\varphi_1 \mid \langle\langle A \rangle\rangle \mathbf{G}\varphi_1 \mid \langle\langle A \rangle\rangle (\varphi_1 \mathbf{U} \varphi_2)$$

where $p \in \mathrm{Prop}$ for a fixed, finite set of proposition symbols Prop, $\bot$ is the false symbol, φ_1 and φ_2 are ATL formulae and $A \subseteq \mathrm{Agt}$ is a subset of a fixed, finite set Agt of players. ATL is the fragment of ATL^* where a temporal quantifier must always be immediately preceeded by a strategy quantifier. This resembles the relationship between CTL and CTL^*. The restrictions give less expressiveness of the logic, but give a lower complexity for the model-checking problem.

We also introduce the fragments ATL_i for $i \geq 0$ of ATL, in which there can only be i nestings of strategic operators. For instance, ATL_0 is the set of

boolean formulae and ATL_1 is the fragment of ATL with no nesting of strategic operators. More formally, ATL_0 formulae are constructed by the grammar

$$\varphi ::= p \mid \bot \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1$$

where $p \in \text{Prop}$ for a fixed, finite set of proposition symbols Prop, $\bot$ is the false symbol and φ_1 and φ_2 are ATL_0 formulae. Inductively, we define the set of ATL_i formulae for $i \geq 1$ by the grammar

$$\varphi ::= p \mid \bot \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \psi_1 \mid \langle\langle A \rangle\rangle \mathbf{X}\psi_1 \mid \langle\langle A \rangle\rangle \mathbf{G}\psi_1 \mid \langle\langle A \rangle\rangle (\psi_1 \mathbf{U} \psi_2)$$

where $p \in \text{Prop}$ for a fixed, finite set of proposition symbols Prop, $\bot$ is the false symbol, φ_1 and φ_2 are ATL_i formulae and ψ_1 and ψ_2 are ATL_{i-1} formulae.

3.2.3 Relationship with other temporal logics

The relationship between the alternating-time logics ATL and ATL^*, the computation tree logics CTL and CTL^* and linear-time temporal logic LTL is illustrated in Figure 3.1 by the partial order $\subseteq$. For two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ we write $\mathcal{L}_1 \subseteq \mathcal{L}_2$ if all formulae of $\mathcal{L}_1$ are also formulae of $\mathcal{L}_2$. There is an edge from $\mathcal{L}_1$ to $\mathcal{L}_2$ if $\mathcal{L}_1 \subseteq \mathcal{L}_2$.

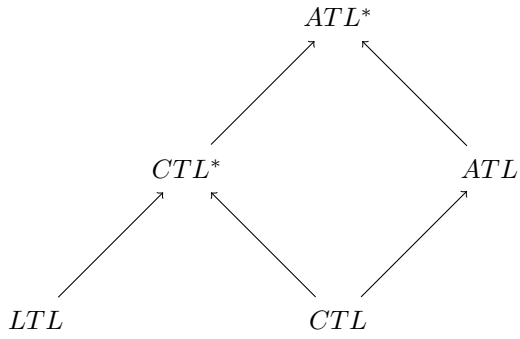

Figure 3.1: Relationship between various temporal logics.

3.3 Semantics

3.3.1 Types of semantics

In the literature there has been introduced semantics for ATL and ATL* with both memoryless strategies and perfect recall strategies [Sch04, LMO08, BLLM09, Jam08, JD08]. This has been done for both complete and incomplete information. We introduce finite-memory semantics for both logics, both in the case of complete and incomplete information. This is motivated by the following two facts

1. In many games memory is needed for winning strategies, which makes memoryless semantics insufficient in some cases. [PR89, DJW97, dAH00, Maz01, AHK02]

2. Model-checking is undecidable for $n \geq 3$ players for perfect recall strategies in ATL/ATL* with incomplete information, even for formulae as simple as $\langle\langle A \rangle\rangle \mathrm{G} p$ where p is a proposition symbol [AHK02, DT11].

The hope is to find decidable and efficiently solvable model-checking problems as in the memoryless case, but with the ability to solve a larger class of games. We consider two new types of semantics. One where the memory size of strategies is bounded by a constant and another where all finite memory strategies are allowed, but not strategies requiring infinite memory. They will be called bounded memory semantics and finite memory semantics respectively. In the litterature i and I are used to denote incomplete and complete information respectively whereas r and R are used to denote memoryless and perfect recall respectively. We will use F_k and F to denote a bounded memory of size k and finite memory respectively. The satisfaction relations will then be denoted $\models_{XY}$ where

- $X \in \{i, I\}$ specifies the type of information available
- $Y \in \{r, F_1, F_2, ..., F, R\}$ specifies the type of strategies available.

For instance, $\mathcal{M}, s \models_{IF_4} \langle\langle\{1\}\rangle\rangle \mathbf{F} p$ means that from state s in CGM $\mathcal{M}$ player 1 has a 4-memory strategy to ensure that p is eventually satisfied. The different types of semantics will be formally defined in the following sections.

3.3.2 Complete Information

The semantics of formulae in alternating-time temporal logic is given with respect to a fixed CGM $\mathcal{M} = (\mathcal{G}, \pi)$ where the players that appear in the formulae must appear in $\mathcal{G}$ and the propositions present in the formulae are in the image Prop of π. For state formulae we define for all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$, all states s, all propositions $p \in \text{Prop}$, all state formulae φ_1 and φ_2, all path formulae φ_3, all coalitions $A \in \text{Agt}$ and all $Y \in \{r, F_1, F_2, ..., F, R\}$

$\mathcal{M}, s \models_{IY} p$	if $p \in \pi(s)$
$\mathcal{M}, s \not\models_{IY} \bot$	
$\mathcal{M}, s \models_{IY} \neg\varphi_1$	if $\mathcal{M}, s \not\models_{IY} \varphi_1$
$\mathcal{M}, s \models_{IY} \varphi_1 \vee \varphi_2$	if $\mathcal{M}, s \models_{IY} \varphi_1$ or $\mathcal{M}, s \models_{IY} \varphi_2$
$\mathcal{M}, s \models_{IY} \langle\langle A \rangle\rangle \varphi_3$	if there exist strategies $(\sigma_A)_{a \in A} \in \prod_{a \in A} \text{Strat}_a^Y$ such that $\forall \rho \in \text{Out}_{\mathcal{G}}(s, \sigma_A). \mathcal{M}, \rho \models_{IY} \varphi_3$

For path formulae we define for all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$, all paths ρ, all propositions $p \in \text{Prop}$, all state formulae φ_1, all path formulae φ_2 and φ_3, all coalitions $A \in \text{Agt}$ and all $Y \in \{r, F_1, F_2, ..., F, R\}$

$\mathcal{M}, \rho \models_{IY} \varphi_1$	if $\mathcal{M}, \rho_0 \models_{IY} \varphi_1$
$\mathcal{M}, \rho \models_{IY} \neg\varphi_2$	if $\mathcal{M}, \rho \not\models_{IY} \varphi_2$
$\mathcal{M}, \rho \models_{IY} \varphi_2 \vee \varphi_3$	if $\mathcal{M}, \rho \models_{IY} \varphi_2$ or $\mathcal{M}, \rho \models_{IY} \varphi_3$
$\mathcal{M}, \rho \models_{IY} \mathbf{X}\varphi_2$	if $\mathcal{M}, \rho_{\geq 1} \models_{IY} \varphi_2$
$\mathcal{M}, \rho \models_{IY} \varphi_2 \mathbf{U} \varphi_3$	if $\exists k. \mathcal{M}, \rho_{\geq k} \models_{IY} \varphi_3$ and $\forall j < k. \mathcal{M}, \rho_{\geq j} \models_{IY} \varphi_2$

The definition of $\models_{IR}$ above is the same as in traditional ATL^* and $\models_{Ir}$ is the memoryless case of complete information as defined in [Sch04]. Note also that the definitions coincide with the definitions for both CTL^* and LTL, where the universal and existential quantifiers $\mathbf{A}$ and $\mathbf{E}$ of CTL^* can be expressed by $\langle\langle \text{Agt} \rangle\rangle$ and $\langle\langle \emptyset \rangle\rangle$ respectively, where Agt is the set of agents in the CGS $\mathcal{G}$.

Consider again the CGM $\mathcal{M}_1 = (\mathcal{G}_1, \pi_1)$ in Figure 3.2.

Let σ_1 be a memoryless strategy for player 1 defined such that $\sigma_1(\rho) = +$ for all ρ with $\text{last}(\rho) = s_1$, such that $\sigma_1(\rho) = -$ for all ρ with $\text{last}(\rho) = s_2$ and defined arbitrarily for other histories. This strategy forces the play to s_3 no matter what strategy player 2 uses. This means that

$$\mathcal{M}_1, s_0 \models_{Ir} \langle\langle \{1\} \rangle\rangle \mathbf{G} \neg p$$

According to our findings in Section 2.3.4 there is a 2-memory strategy for player 1 to make sure that both s_5 and s_6 is visited infinitely often, but no memoryless

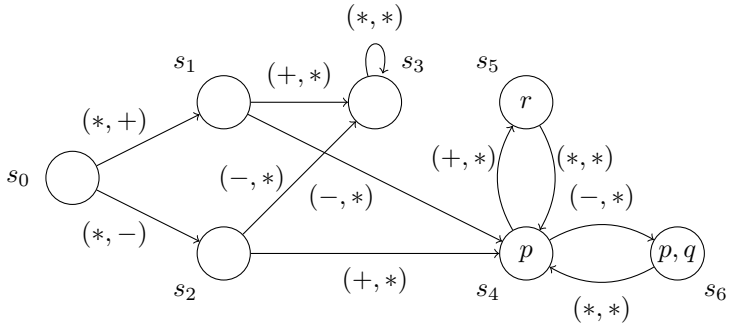

Figure 3.2: Concurrent game model $\mathcal{M}_1 = (\mathcal{G}_1, \pi_1)$

strategy to do so. This means that

$$\mathcal{M}_1, s_0 \not\models_{Ir} \langle\langle\{1\}\rangle\rangle(\mathbf{GF}r \wedge \mathbf{GF}q)$$

$$\mathcal{M}_1, s_0 \models_{IF_2} \langle\langle\{1\}\rangle\rangle(\mathbf{GF}r \wedge \mathbf{GF}q)$$

$$\mathcal{M}_1, s_0 \models_{IF} \langle\langle\{1\}\rangle\rangle(\mathbf{GF}r \wedge \mathbf{GF}q)$$

$$\mathcal{M}_1, s_0 \models_{IR} \langle\langle\{1\}\rangle\rangle(\mathbf{GF}r \wedge \mathbf{GF}q)$$

In general we have that if a coalition has a memoryless strategy to win with some objective, then it has a k-memory strategy for any k which can be obtained by adding disconnected memory states to the Mealy automaton representing the strategy. Existence of a k-memory winning strategy immediately implies a finite-memory winning strategy, since any k-memory strategy is a finite-memory strategy. Existence of a finite-memory winning strategy also immediately implies existence of a perfect recall winning strategy, since any finite-memory strategy is also a perfect recall strategy. In the next chapter we will explore in more detail the differences between the different types of semantics in ATL and ATL^* with both complete and incomplete information.

3.3.3 Incomplete Information

As a generalization of the complete information case, we define the semantics for ATL/ATL^* for incomplete information. Here, the semantics is given with respect to a fixed iCGM $\mathcal{M} = (\mathcal{G}, \pi)$ and we still require that players appearing in the formulae must appear in $\mathcal{G}$ and the propositions present in the formulae

are in the image Prop of π. Now, for state formulae we define for all iCGMs $\mathcal{M} = (\mathcal{G}, \pi)$, all states s, all propositions $p \in \text{Prop}$, all state formulae φ_1 and φ_2, all path formulae φ_3, all coalitions $A \in \text{Agt}$ and all $Y \in \{r, F_1, F_2, ..., F, R\}$

$\mathcal{M}, s \models_{iY} p$	if $p \in \pi(s)$
$\mathcal{M}, s \not\models_{iY} \bot$	
$\mathcal{M}, s \models_{iY} \neg\varphi_1$	if $\mathcal{M}, s \not\models_{iY} \varphi_1$
$\mathcal{M}, s \models_{iY} \varphi_1 \vee \varphi_2$	if $\mathcal{M}, s \models_{iY} \varphi_1$ or $\mathcal{M}, s \models_{iY} \varphi_2$
$\mathcal{M}, s \models_{iY} \langle\langle A \rangle\rangle \varphi_3$	if there exist strategies $(\sigma_A)_{a \in A} \in \prod_{a \in A} \text{Strat}_a^Y$ such that for every $a \in A$, every $s' \sim_a s$ and every $\rho \in \text{Out}_{\mathcal{G}}(s', \sigma_A)$ we have $\mathcal{M}, \rho \models_{iY} \varphi_3$

For path formulae we define for all iCGMs $\mathcal{M} = (\mathcal{G}, \pi)$, all paths ρ, all propositions $p \in \text{Prop}$, all state formulae φ_1, all path formulae φ_2 and φ_3, all coalitions $A \in \text{Agt}$ and all $Y \in \{r, F_1, F_2, ..., F, R\}$

$\mathcal{M}, \rho \models_{iY} \varphi_1$	if $\mathcal{M}, \rho_0 \models_{iY} \varphi_1$
$\mathcal{M}, \rho \models_{iY} \neg\varphi_2$	if $\mathcal{M}, \rho \not\models_{iY} \varphi_2$
$\mathcal{M}, \rho \models_{iY} \varphi_2 \vee \varphi_3$	if $\mathcal{M}, \rho \models_{iY} \varphi_2$ or $\mathcal{M}, \rho \models_{iY} \varphi_3$
$\mathcal{M}, \rho \models_{iY} \mathbf{X}\varphi_2$	if $\mathcal{M}, \rho_{\geq 1} \models_{iY} \varphi_2$
$\mathcal{M}, \rho \models_{iY} \varphi_2 \mathbf{U} \varphi_3$	if $\exists k. \mathcal{M}, \rho_{\geq k} \models_{iY} \varphi_3$ and $\forall j < k. \mathcal{M}, \rho_{\geq j} \models_{iY} \varphi_2$

The main difference between the incomplete information and complete information case is in the strategy quantifiers. Since players don't have complete information they can only choose from uniform strategies. In addition, if the players in a coalition A have a strategy to make sure that some property φ is satisfied from some state s in iCGM $\mathcal{M}$, then we only write $\mathcal{M}, s \models_{iY} \langle\langle A \rangle\rangle \varphi$ for some Y if this strategy also satisfies the property if the play starts in states that are indistinguishable to s for one or more of the players in A. Defining it as in the complete information case is also a valid option and was also done in some other works, e.g. [DT11, ÅW09]. However, this corresponds in some sense to letting the players have complete information about the initial state which we feel is less natural. The definition we choose is also used in most cases, e.g. in [Sch04, JD08, BDJ10, BGJ12].

Consider again the iCGM $\mathcal{M}_2 = (\mathcal{G}_2, \pi_2)$ in Figure 3.3. For this game, none of the formulae presented in the examples on complete information semantics are true in s_0. This is because there is neither a strategy for player 1 to force the play to s_3 nor a strategy for player 1 to force the play to s_4, since he has to choose the same action in both s_1 and s_2. Thus,

$$\mathcal{M}_2, s_0 \not\models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{G} \neg p$$

$$\mathcal{M}_2, s_0 \not\models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{F} p$$

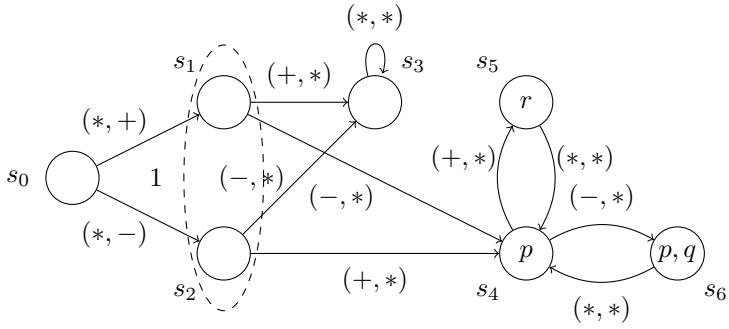

Figure 3.3: iCGM $\mathcal{M}_2 = (\mathcal{G}_2, \pi_2)$

However, this does not mean that player 2 can enforce the play to s_3 or enforce the play to s_4 starting in s_0, because no matter which strategy player 2 chooses, player 1 can choose a spoiling strategy to go the other way. Thus,

$$\mathcal{M}_2, s_0 \not\models_{iR} \langle\langle\{2\}\rangle\rangle \mathbf{G}\neg p$$

$$\mathcal{M}_2, s_0 \not\models_{iR} \langle\langle\{2\}\rangle\rangle \mathbf{F}p$$

However, if the two players work together they can enforce the play to s_3 or enforce the play to s_4 starting in s_0, since they can choose a collective strategy σ so all outcomes of σ pass through the state they wish. This can even be done by memoryless strategies and thus

$$\mathcal{M}_2, s_0 \models_{ir} \langle\langle\{1,2\}\rangle\rangle \mathbf{G}\neg p$$

$$\mathcal{M}_2, s_0 \models_{ir} \langle\langle\{1,2\}\rangle\rangle \mathbf{F}p$$

Another interesting formula to look at is $\langle\langle\{2\}\rangle\rangle\langle\langle\{1\}\rangle\rangle \mathbf{F}p$ stating that player 2 has a strategy such that player 1 can have a strategy that makes sure the play reaches s_4 when starting the play in s_0. A bit surprisingly however, we have

$$\mathcal{M}_2, s_0 \not\models_{iR} \langle\langle\{2\}\rangle\rangle\langle\langle\{1\}\rangle\rangle \mathbf{F}p$$

Let us see why. Assume for contradiction that it is not the case, i.e. that the formula is true in s_0. According to the definition of the semantics we have

$$\mathcal{M}_2, s_0 \models_{iR} \langle\langle\{2\}\rangle\rangle\langle\langle\{1\}\rangle\rangle \mathbf{F}p \text{ if and only if}$$

$$\exists \sigma_2 \in \text{Strat}_2^R . \forall \rho \in \text{Out}_{\mathcal{G}}(s, \sigma_2) . \mathcal{M}_2, \rho \models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{F} p$$

since s_0 is only indistinguishable to itself from the point of view of player 2. Since $\langle\langle \{1\} \rangle\rangle \mathbf{F} p$ is a state formula, the above is equivalent to

$$\exists \sigma_2 \in \text{Strat}_2^R . \forall \rho \in \text{Out}_{\mathcal{G}}(s, \sigma_2) . \mathcal{M}_2, \rho_0 \models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{F} p$$

Now, for every outcome ρ of a given strategy σ_2 of player 2, we have $\rho_0 = s_0$. Thus, the above implies

$$\exists \sigma_2 \in \text{Strat}_2^R . \forall \rho \in \text{Out}_{\mathcal{G}}(s, \sigma_2) . \mathcal{M}_2, s_0 \models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{F} p \text{ implies}$$

$$\mathcal{M}_2, s_0 \models_{iR} \langle\langle \{1\} \rangle\rangle \mathbf{F} p$$

But we already know that this is false, so we have a contradiction. The intuition is that player 1 cannot be sure to reach s_4 from s_0, even if player 2 says he will follow a certain strategy, because player 1 cannot be sure that player 2 actually commits to the strategy. What a strategy quantifier $\langle\langle A \rangle\rangle$ does in a formula $\langle\langle A \rangle\rangle \varphi$ is to restrict the set of outcomes considered when evaluating φ. However, in either of the two sets of outcomes that player 2 can choose between by selecting $+$ or $-$, player 1 cannot choose a strategy that is sure to enforce the play to go to s_4 from s_0. It has been argued by some authors (e.g.[ÅW09, BLLM09, BGJ12]), that there is also the need of logics where players are required to commit to the strategies specified by the strategy quantifiers. A variant of ATL/ATL^* has been developed, which is called ATL/ATL^* with strategy context and bounded memory [BLLM09]. In this logic, it is possible to have strategy quantifiers where players have to commit to the strategy specified by the quantifier unlike in ATL/ATL^*. We will not go into more detail with this logic as it is out of the scope of this thesis.

3.4 Other variants of $ATL/ATL*$

In addition to ATL with strategy context and bounded memory [BLLM09] some other variants of ATL/ATL^* have been defined in the litterature. This include Fair ATL [AHK02] which is motivated by the use of fairness constraints in CTL and probabilistic ATL/ATL^* [CL07] which generalize probabilistic CTL/CTL^*. There has also been developed a logic called ATL with bounded memory [ÅW09] in which players are only allowed to use information about a constant length suffix of the history of the game, which gives a different logic from what we consider in this thesis, since we let players remember the parts of the history they want, but where they can only store a certain amount of memory.

Chapter 4

ATL/ATL^* Expressiveness

4.1 Overview

In this chapter we explore differences in expressiveness between the different types of semantics introduced in the previous chapter. This is done to show that they are indeed all unique and that none of them are superfluous. In addition, the insights gained by analyzing the difference in expressiveness in this chapter will be exploited when developing algorithms for model-checking in the next chapter. Some of the results proved in this chapter can be done in a shorter and more high-level manner, but without providing as much intuition about the details and inner workings of the games considered. We have chosen this path, both in order to illustrate properties of concurrent games as well as to obtain intermediate results that can be used in the model-checking chapter.

The main results in this chapter are shown in Figure 4.1. Here, we denote by $\mathbb{M}^{\mathcal{L}}_{XY}$ the set of 3-tuples $(\mathcal{M}, \rho, \varphi)$ such that $\mathcal{M}, \rho \models_{XY} \varphi$ where $\mathcal{M}$ is a (i)CGM, ρ is a path in $\mathcal{M}$, φ is a formula in $\mathcal{L} \in \{ATL, ATL^*\}$, $X \in \{i, I\}$ and $Y \in \{r, F_1, F_2, ..., F, R\}$.

In ATL with complete information, no memory is needed. This is roughly because all ATL formulae are state formulae and are therefore either true or not true in a given state. We will present a construction which relates the winning

Logic	Expressiveness
ATL w. complete info	$\mathbb{M}_{Ir}^{ATL} = \mathbb{M}_{IF_2}^{ATL} = \mathbb{M}_{IF_3}^{ATL} ... = \mathbb{M}_{IF}^{ATL} = \mathbb{M}_{IR}^{ATL}$
ATL w. incomplete info	$\mathbb{M}_{ir}^{ATL} \subset \mathbb{M}_{iF_2}^{ATL} \subset \mathbb{M}_{iF_3}^{ATL} ... \subset \mathbb{M}_{iF}^{ATL} \subseteq \mathbb{M}_{iR}^{ATL}$
ATL^* w. complete info	$\mathbb{M}_{Ir}^{ATL^*} \subset \mathbb{M}_{IF_2}^{ATL^*} \subset \mathbb{M}_{IF_3}^{ATL^*} ... \subset \mathbb{M}_{IF}^{ATL^*} = \mathbb{M}_{IR}^{ATL^*}$
ATL^* w. incomplete info	$\mathbb{M}_{ir}^{ATL^*} \subset \mathbb{M}_{iF_2}^{ATL^*} \subset \mathbb{M}_{iF_3}^{ATL^*} ... \subset \mathbb{M}_{iF}^{ATL^*} \subset \mathbb{M}_{iR}^{ATL^*}$

Figure 4.1: Relations between the different types of semantics

strategies for coalitions of players in complete information concurrent games to winning strategies in 2-player turn-based games. In the case of ATL the winning strategies will be closely related to winning strategies in 2-player turn-based games with reachability and safety objectives, where existence of a winning strategy is equivalent to existence of a memoryless winning strategy. This was shown for parity conditions, generalizing reachability and safety conditions in [EJ91]. The same construction will be used for ATL^* in the case of complete information. Here, the winning strategies will be closely related to winning strategies in 2-player turn-based games with LTL objectives, where memoryless strategies are not enough, but where existence of a winning strategy is equivalent to existence of a finite-memory strategy [PR89]. We will show that the bounded-memory hierarchies are strict for ATL^* both with complete and incomplete information and for ATL with incomplete information. This will be done by using simple examples illustrating the difference. In the chapter on model-checking we will also prove that for ATL^* with incomplete information infinite memory is actually needed unlike in the complete information case. However, for ATL with incomplete information we do not yet know whether the inclusion $\mathbb{M}_{iF}^{ATL} \subseteq \mathbb{M}_{iR}^{ATL}$ is strict.

4.2 Strictness of bounded-memory hierarchies

We start by showing strictness of the bounded-memory hierarchy for incomplete information games

Proposition 6 $\mathbb{M}_{iF_k}^{\mathcal{L}} \subset \mathbb{M}_{iF_{k+1}}^{\mathcal{L}}$ *for all* $k \geq 1$ *and* $\mathcal{L} \in \{ATL, ATL^*\}$

Proof. For all iCGMs $\mathcal{M}$, all paths ρ, every ATL^* formula φ and all $k \geq 1$ we have that $\mathcal{M}, \rho \models_{iF_k} \varphi$ implies $\mathcal{M}, \rho \models_{iF_{k+1}} \varphi$. The reason is that if a coalition has a k-memory strategy σ_k to satisfy some property, then a $k+1$-memory

strategy σ_{k+1} can be obtained by adding a disconnected node to the Mealy automata representing the strategies in σ_k. Then the two strategies will behave in the same way.

We will now show that for all $k \geq 2$ there exists an iCGM $\mathcal{M}_k = (\mathcal{G}_k, \pi_k)$, an initial state s_0 and an ATL formula φ such that $\mathcal{M}_k, s_0 \models_{iF_k} \varphi$ and $\mathcal{M}_k, s_0 \not\models_{iF_{k-1}} \varphi$. Consider the one-player game in Figure 4.2 where player 1 cannot distinguish between any of the states and his goal is to eventually reach s_{win}.

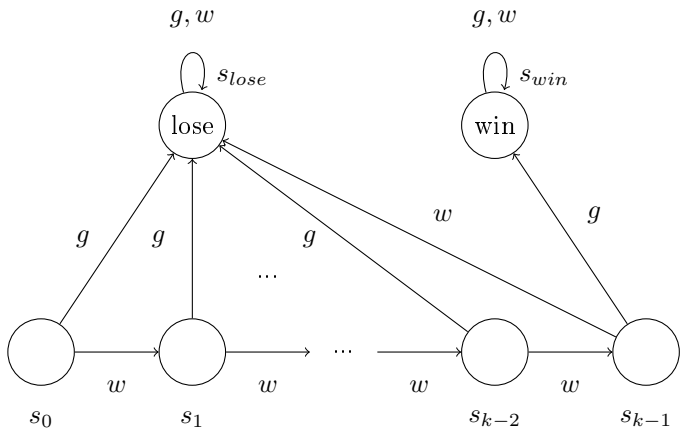

Figure 4.2: iCGM $\mathcal{M}_k$

All states are contained in the information set I_1 and the player wins the game if he chooses w (wait) in the first $k-1$ rounds and then chooses g (go) in the kth round. The idea is that he has to remember how many times he has already chosen to wait in order to decide when to go, because he cannot see which state he is in at any point in the game.

We will show that $\mathcal{M}_k, s_0 \models_{iF_k} \langle\langle\{1\}\rangle\rangle \mathbf{F}\text{win}$, but also that $\mathcal{M}_k, s_0 \not\models_{iF_{k-1}} \langle\langle\{1\}\rangle\rangle \mathbf{F}\text{win}$.

First, let σ be a k-memory strategy defined by the Mealy automaton $\mathcal{A} = (M, m_0, \{I_1\}, \{w, g\}, T, G)$ where $|M| = k$ which is illustrated in Figure 4.3.

This strategy is winning for player 1, since it will choose to wait for the first $k-1$ rounds and then choose to go. Thus, $\mathcal{M}_k, s_0 \models_{iF_k} \langle\langle\{1\}\rangle\rangle \mathbf{F}\text{win}$.

For contradiction, suppose there is a $k-1$-memory strategy σ' for player 1 to win the game. Let it be represented by the Mealy automaton $\mathcal{A}' = (M', m_0', \{I_1\}, \{w, g\}, T', G')$.

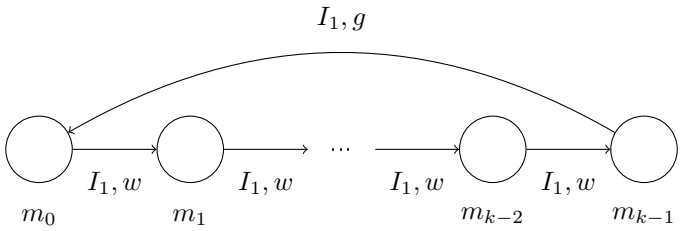

Figure 4.3: Mealy automaton $\mathcal{A}$ representing σ

The only possible winning outcome of the game is $s_0 s_1 ... s_{k-1} s_{win}^{\omega}$. Let the sequence m' of memory-states in $\mathcal{A}'$ when the game is played using σ' be

$$m' = m'_0 m'_1 ...$$

where m'_0 is the initial memory and m'_i is the memory after the first i steps has been played. Since there are only $k-1$ different memory states in $\mathcal{A}'$, there must be a repeated memory state in the prefix $m'_0 m'_1 ... m'_{k-1}$ of m'. Let j be the smallest index such that m'_j appears in $m'_0 m'_1 ... m'_{j-1}$. Then $j \leq k-1$. Since the Mealy automaton only has one input symbol, it will continue to loop such that

$$m' = m'_0 ... m'_{p-1} (m'_p m'_{p+1} ... m'_{j-1})^{\omega}$$

where m'_p is the first memory state such that $m'_p = m'_j$. The corresponding sequence a of actions played by player 1 is then

$$a = G'(m'_0, I_1) ... G'(m'_{p-1}, I_1)(G'(m'_p, I_1) G'(m'_{p+1}, I_1) ... G'(m'_{j-1}, I_1))^{\omega}$$

Player 1 does not choose to go during the first $k-1$ rounds since σ' is winning. But according to the action sequence a played by player 1, the set of actions he chooses from during the game is the same as the set of actions he chooses from during the first $j \leq k-1$ rounds, which means that he never chooses to go. This means that σ' is not winning which gives a contradiction and therefore no winning $k-1$-memory strategy exists for player 1. Thus $\mathcal{M}_k, s_0 \not\models_{iF_{k-1}} \langle\langle\{1\}\rangle\rangle \mathbf{F}\text{win}$. □

Next, we prove the strictness for ATL^* with complete information in a similar game.

Proposition 7 $\mathbb{M}^{ATL^*}_{IF_k} \subset \mathbb{M}^{ATL^*}_{IF_{k+1}}$ *for all* $k \geq 1$.

Proof. Because $\mathbb{M}^{ATL^*}_{iF_k} \subset \mathbb{M}^{ATL^*}_{iF_{k+1}}$ we have that $\mathbb{M}^{ATL^*}_{IF_k} \subseteq \mathbb{M}^{ATL^*}_{IF_{k+1}}$. To show strictness, we consider a one-player CGM $\mathcal{M}$ where the player has a family of objectives $\varphi_k = \mathbf{X}^k p$ which gives a similar family of games to the one we considered for the incomplete information case. $\mathcal{M}$ is illustrated in Figure 4.4.

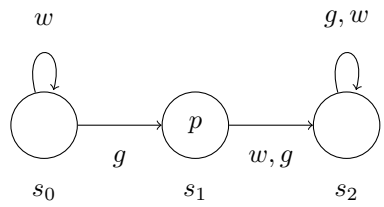

Figure 4.4: CGM $\mathcal{M}$

In this one-player game, the objective φ_k of player 1 is to choose w (wait) for the first $k-1$ rounds and g (go) in the kth round. We can now use the same arguments is in the proof of Proposition 6 because the actions chosen by player 1 when the play is in s_1 and s_2 does not influence the outcome of the game. This means that the Mealy automaton that represents a finite-memory strategy of the player will receive the same input symbol for as long as player 1 still has any choice that affects the outcome and therefore the game is essentially the same with the given objectives. Thus, we have $\mathcal{M}, s_0 \models_{IF_k+1} \mathbf{X}^{k+1} p$ and $\mathcal{M}, s_0 \not\models_{IF_k} \mathbf{X}^{k+1} p$ for all $k \geq 1$ and thereby $\mathbb{M}^{ATL^*}_{IF_k} \subset \mathbb{M}^{ATL^*}_{IF_{k+1}}$ for all $k \geq 1$. □

Note that the models used in the two previous proofs also show that there is no constant bound on the memory needed in either ATL with incomplete memory or ATL^* with complete or incomplete information. This gives us

Corollary 8 $\mathbb{M}^{ATL}_{iF_k} \subset \mathbb{M}^{ATL}_{iF}$, $\mathbb{M}^{ATL^*}_{iF_k} \subset \mathbb{M}^{ATL^*}_{iF}$ *and* $\mathbb{M}^{ATL^*}_{IF_k} \subset \mathbb{M}^{ATL^*}_{IF}$ *for all* $k \geq 1$.

4.3 A reduction to turn-based games

As a first step to proving the expressiveness results for complete information game structures, we present a construction of a 2-player turn-based CGM $\mathcal{M}_A = (\mathcal{G}_A, \pi')$ from an arbitrary CGM $\mathcal{M} = (\mathcal{G}, \pi)$ and coalition A of players. The idea is that coalition A has a winning strategy in $\mathcal{M}$ from state s with LTL objective φ if and only if player 1 has a winning strategy in $\mathcal{M}_A$ from state s with another, but similar, LTL objective φ'. In addition, this correspondence holds for finite-memory winning strategies and memoryless winning strategies as well. The construction of the turn-based game is such that player 1 controls the players in coalition A and player 2 controls the players in $\mathrm{Agt} \setminus A$. In addition, player 2 gets to see which choices player 1 makes in each round before having to choose his own move. The construction is illustrated in Figure 4.5 and is defined more formally as follows.

First, let $\mathrm{Agt} = \{1, ..., n\}$ and assume without loss of generality that A consists of the first N players, where $N \leq n$, i.e. $A = \{1, ..., N\}$. Let $\mathcal{G} = (\mathrm{States}, \mathrm{Agt}, \mathrm{Act}, \mathrm{Mov}, \mathrm{Tab})$. Then define $\mathcal{G}_A = (\mathrm{States}', \mathrm{Agt}', \mathrm{Act}', \mathrm{Mov}', \mathrm{Tab}')$ where

- $\mathrm{States}' = \mathrm{States} \cup \{s_m | s \in \mathrm{States} \wedge m \in \prod_{j=1}^{N} \mathrm{Mov}(s, j)\}$
- $\mathrm{Agt}' = \{1, 2\}$
- $\mathrm{Act}' = \mathrm{Act}^N \cup \mathrm{Act}^{n-N} \cup \{*\}$
- $\mathrm{Mov}'(s, 1) = \prod_{j=1}^{N} \mathrm{Mov}(s, j)$ for all $s \in \mathrm{States}$
- $\mathrm{Mov}'(s, 2) = \{*\}$ for all $s \in \mathrm{States}$
- $\mathrm{Mov}'(s_m, 1) = \{*\}$ for all $s_m \in \mathrm{States}' \setminus \mathrm{States}$
- $\mathrm{Mov}'(s_m, 2) = \prod_{j=N+1}^{n} \mathrm{Mov}(s, j)$ for all $s_m \in \mathrm{States}' \setminus \mathrm{States}$
- $\mathrm{Tab}'(s, (m, *)) = s_m$ for all $s \in \mathrm{States}$ and $m \in \mathrm{Mov}'(s, 1)$
- $\mathrm{Tab}'(s_m, (*, m')) = \mathrm{Tab}(s, (m, m'))$ for all $s_m \in \mathrm{States}' \setminus \mathrm{States}$ and $m' \in \mathrm{Mov}'(s_m, 2)$

and let $\pi'(s) = \pi(s)$ for all $s \in \mathrm{States}$ and $\pi'(s_m) = \pi(s)$ for all $s_m \in \mathrm{States}' \setminus \mathrm{States}$. The idea is to let player 1 control coalition A and let player 2 control coalition $\mathrm{Agt} \setminus A$. In addition, each round in $\mathcal{G}$ corresponds to two rounds in $\mathcal{G}_A$. One round where player 1 chooses actions for players in A and another round where player 2 chooses actions for players in $\mathrm{Agt} \setminus A$ with the knowledge of the

choice of player 1. In Figure 4.5, player 1 in $\mathcal{M}_A$ controls the first player in $\mathcal{M}$ and player 2 always gets to know what player 1 chooses before he chooses an action himself. The newly added states are drawn dashed. We claim that in general, this construction provides a connection between sure-winning strategies of coalition A in $\mathcal{M}$ and sure-winning strategies of player 1 in $\mathcal{M}_A$.

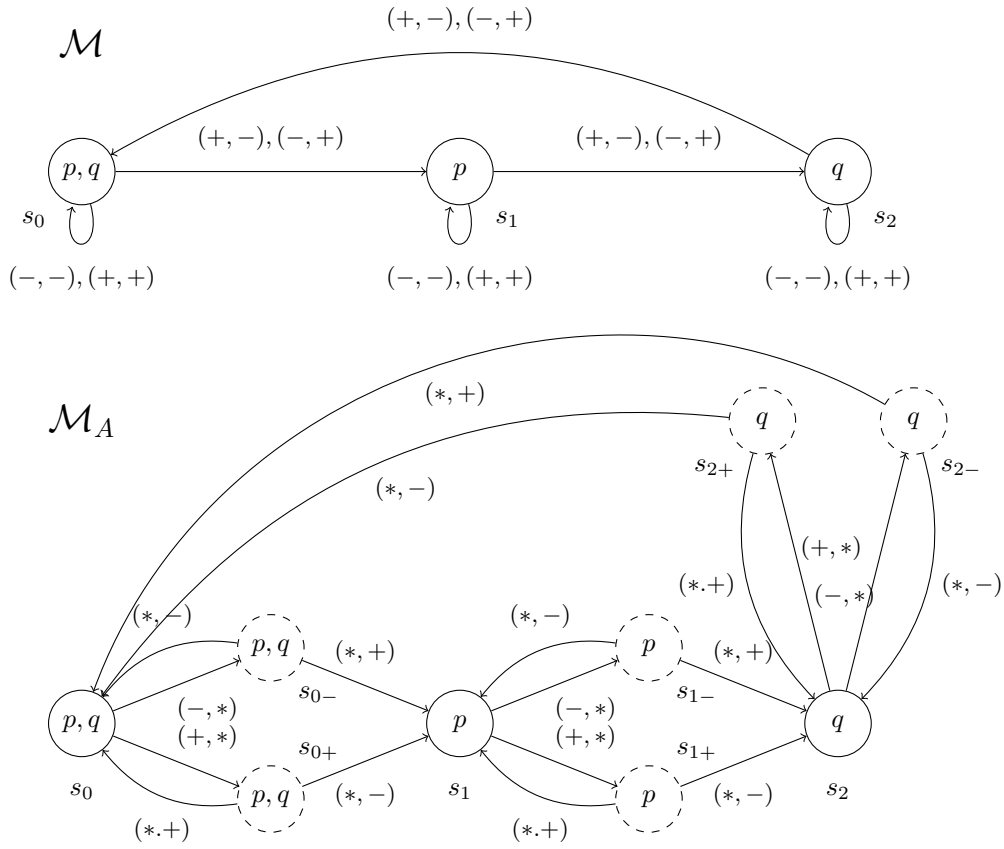

Figure 4.5: Construction of $\mathcal{M}_A$ from $\mathcal{M}$ where $A = \{1\}$.

To show the correspondence between $\mathcal{M}$ and $\mathcal{M}_A$, the following Lemma (which is proven in Appendix A) is needed, since every step of the concurrent game corresponds to two steps of the turn-based game.

Lemma 9 *For all LTL formulae φ and $\varphi' = \varphi[\boldsymbol{X} \mapsto \boldsymbol{XX}]$ over some finite alphabet Σ it holds that for all sequences $a_1a_2a_3... \in (2^\Sigma)^\omega$*

$$a_1a_2a_3... \models_{LTL} \varphi \text{ iff } a_1a_1a_2a_2a_3a_3... \models_{LTL} \varphi'$$

Next, we show that coalition A has a sure-winning strategy in a CGM $\mathcal{M} = (\mathcal{G}, \pi)$ from state s with LTL objective φ if and only if player 1 has a sure-winning strategy in the turn-based CGM $\mathcal{M}_\mathcal{A} = (\mathcal{G}_A, \pi')$ from state s with LTL objective $\varphi' = \varphi[\mathbf{X} \mapsto \mathbf{XX}]$. Moreover, finite memory is needed to sure-win in the first game if and only if it is in the second.

Lemma 10 *For all LTL formulae φ, all concurrent game structures $\mathcal{M} = (\mathcal{G}, \pi)$ and all states s we have*

1. $\mathcal{M}, s \models_{IR} \langle\langle A \rangle\rangle \varphi$ *if and only if* $\mathcal{M}_A, s \models_{IR} \langle\langle 1 \rangle\rangle \varphi[\boldsymbol{X} \mapsto \boldsymbol{XX}]$
2. $\mathcal{M}, s \models_{IF} \langle\langle A \rangle\rangle \varphi$ *if and only if* $\mathcal{M}_A, s \models_{IF} \langle\langle 1 \rangle\rangle \varphi[\boldsymbol{X} \mapsto \boldsymbol{XX}]$
3. $\mathcal{M}, s \models_{Ir} \langle\langle A \rangle\rangle \varphi$ *if and only if* $\mathcal{M}_A, s \models_{Ir} \langle\langle 1 \rangle\rangle \varphi[\boldsymbol{X} \mapsto \boldsymbol{XX}]$

Proof. We start by proving the lemma for perfect recall strategies (case 1.).

($\Rightarrow$) For the first direction, let coalition A have a sure-winning strategy σ in $\mathcal{G}$ from s. Now define for player 1 in $\mathcal{G}_A$ a corresponding strategy σ^1 such that

$$\sigma^1(s_1 s_{1m_1} s_2 s_{2m_2} ... s_k) = (\sigma_1(s_1 s_2 ... s_k), ..., \sigma_N(s_1 s_2 ... s_k))$$

for all histories $s_1 s_{1m_1} s_2 s_{2m_2} ... s_k$ in $\mathcal{G}_A$ where player 1 has another action available than $*$.

Assume for contradiction that σ^1 is not sure-winning for player 1 in $\mathcal{G}_A$ from s with objective $\varphi' = \varphi[\mathbf{X} \mapsto \mathbf{XX}]$. Then there exists a spoiling strategy σ^2 for player 2 in $\mathcal{G}_A$ such that $\text{Out}_{\mathcal{G}_A}(s, \sigma^1, \sigma^2) \not\models \varphi'$. Denote $\text{Out}_{\mathcal{G}_A}(s, \sigma^1, \sigma^2) = t_1 t_{1m_1} t_2 t_{2m_2} ....$ From σ^2 we define a strategy σ' for the coalition $\text{Agt} \setminus A$ as follows

$$\sigma'_j(t_1 t_2 ... t_k) = (\sigma^2(t_1 t_{1m_1} t_2 t_{2m_2} ... t_k t_{km_k}))_{j-N} \text{ for all } k \geq 1 \text{ and all } j > N$$

for the plays "following" the outcome in $\mathcal{G}_A$ for σ^1 and σ^2 and arbitrarily for all other histories of play. We now have that

$$\text{Out}_\mathcal{G}(s, \sigma, \sigma') = t_1 t_2 ...$$

since

$$\begin{aligned}
&\mathrm{Tab}(t_j, (\sigma_1(t_{\leq j}), ..., \sigma_N(t_{\leq j}), \sigma'_{N+1}(t_{\leq j}), ..., \sigma'_n(t_{\leq j})))\\
&= \mathrm{Tab}'(t_{j(\sigma_1(t_{\leq j}),...,\sigma_N(t_{\leq j}))}, (*, (\sigma'_{N+1}(t_{\leq j}), ..., \sigma'_n(t_{\leq j}))))\\
&= \mathrm{Tab}'(t_{j\sigma^1(t_1 t_{1m_1}...t_j)}, (*, \sigma^2(t_1 t_{1m_1}...t_j t_{jm_j})))\\
&= \mathrm{Tab}'(t_{jm_j}, (*, \sigma^2(t_1 t_{1m_1}...t_j t_{jm_j})))\\
&= t_{j+1}
\end{aligned}$$

for all $j \geq 1$.

Since player 1 loses in the play $t_1 t_{m_1} t_2 t_{m_2}...$ we have $t_1 t_{m_1} t_2 t_{m_2}... \not\models \varphi'$, but since $\pi'(t_{jm_j}) = \pi'(t_j)$ for $j \geq 1$ we have $t_1 t_1 t_2 t_2... \not\models \varphi'$ and by Lemma 9 that $t_1 t_2... = \mathrm{Out}_{\mathcal{G}}(s, \sigma, \sigma') \not\models \varphi$ which means that σ' is a spoiling strategy for σ. This gives a contradiction since σ is sure-winning for A in $\mathcal{G}$ from s. Thus the strategy σ^1 must be sure winning in $\mathcal{G}_A$ from s.

($\Leftarrow$) For the second direction, let player 1 have a sure-winning strategy σ^1 in $\mathcal{G}_A$ from s with objective $\varphi' = \varphi[\mathbf{X} \mapsto \mathbf{XX}]$. From this we design a strategy σ for the coalition A in $\mathcal{G}$ by

$$\sigma_\ell(s_1 s_2 ... s_k) = \sigma^1(s_1 s_{1m_1} s_2 s_{2m_2} ... s_k)_\ell$$

for all $1 \leq \ell \leq N$ and all sequences $s_1, ..., s_k \in \mathrm{Hist}_{\mathcal{G}}$ where $m_j = \sigma^1(s_1 s_{1m_1} s_2 s_{2m_2} ... s_j)$ for all j.

Assume for contradiction that σ is not sure-winning in $\mathcal{G}$ from s with objective σ. Then there exists a spoiling strategy σ' for $\mathrm{Agt} \setminus A$ such that

$$\mathrm{Out}_{\mathcal{G}}(s, (\sigma, \sigma')) = t_1 t_2... \not\models \varphi$$

From σ' we define a strategy σ^2 for player 2 in $\mathcal{G}_A$ for all k by

$$\sigma^2(t_1 t_{1M_1} t_2 t_{2M_2} ... t_k t_{kM_k})_{\ell - N} = \sigma'_\ell(t_1 t_2 ... t_k)$$

for all $N + 1 \leq \ell \leq n$ where $M_j = \sigma^1(t_1 t_{1M_1} ... t_j)$ for all j. σ^2 is defined arbitrarily for all other sequences that don't "follow" the outcome of σ and σ' in $\mathcal{G}$.

We now have that $\mathrm{Out}_{\mathcal{G}_A}(s,(\sigma^1,\sigma^2)) = t_1 t_{1M_1} t_2 t_{2M_2}...$ since for all $j \geq 1$

$$\mathrm{Tab}'(t_j,(\sigma^1(t_1 t_{1M_1}...t_j),*)) = \mathrm{Tab}'(t_j,(M_j,*)) = t_{jM_j} \text{ and}$$

$$\mathrm{Tab}'(t_{jM_j},(*,\sigma^2(t_1 t_{1M_1}...t_j t_{jM_j})))$$

$$= \mathrm{Tab}'(t_{jM_j},(*,(\sigma'_{N+1}(t_1 t_2...t_j),...,\sigma'_n(t_1 t_2...t_j))))$$

$$= \mathrm{Tab}(t_j,(M_j,(\sigma'_{N+1}(t_1 t_2...t_j),...,\sigma'_n(t_1 t_2...t_j))))$$

$$= \mathrm{Tab}(t_j,(\sigma_1(t_1 t_2...t_j),...,\sigma_N(t_1 t_2...t_j),\sigma'_{N+1}(t_1 t_2...t_j),...,\sigma'_n(t_1 t_2...t_j)))$$

$$= t_{j+1}$$

Since $t_1 t_2... \not\models \varphi$ we have $t_1 t_1 t_2 t_2... \not\models \varphi'$ according to Lemma 9. And since $\pi'(t_\ell) = \pi'(t_{\ell M})$ for every $\ell \geq 1$ and every move M we have $\mathrm{Out}_{\mathcal{G}_A}(s,(\sigma^1,\sigma^2)) = t_1 t_{1M_1} t_2 t_{2M_2}... \not\models \varphi'$. Thus, σ^2 is a spoiling strategy for σ^1 from s. But this gives a contradiction since σ^1 is winning in $\mathcal{G}_A$ from s with objective φ' and thus σ must be winning in $\mathcal{G}$ from s with objective φ. This finishes the Lemma for perfect recall strategies.

For case 2. with finite-memory semantics, the proof is the same as in case 1. However, with the first direction ($\Rightarrow$) we need to show that if σ is a finite-memory strategy, then so is σ^1. For the second direction ($\Leftarrow$) we need to show that if σ^1 is a finite-memory strategy, then so is σ.

($\Rightarrow$) For the first direction, we assume σ_j is given by the Mealy automata $\mathcal{A}_j = (W_j, w_{0j}, \mathrm{States}, \mathrm{Act}, T_j, G_j)$. Then define σ^1 by the Mealy automaton $\mathcal{A} = (W, w_0, \mathrm{States}', \mathrm{Act}', T, G)$ where

- $W = \prod_{j=1}^N W_j$
- $w_0 = (w_{01},...,w_{0N})$
- $\mathrm{States}' = \mathrm{States} \cup \{s_m | s \in \mathrm{States}, m \in \mathrm{Act}^N, m_j \in \mathrm{Mov}(s,j) \text{ for } 1 \leq j \leq N\}$
- $\mathrm{Act}' = \mathrm{Act}^N \cup \{*\}$
- $T'((w_1,...,w_N),s) = (w_1,...,w_N)$ if $s \notin \mathrm{States}$
- $T'((w_1,...,w_N),s) = (T_1(w_1,s),...,T_n(w_N,s))$ if $s \in \mathrm{States}$
- $G'((w_1,...,w_N),s) = *$ if $s \notin \mathrm{States}$

- $G'((w_1, ..., w_N), s) = (G_1(w_1, s), ..., G_N(w_N, s))$ if $s \in \text{States}$

This indeed is a finite-memory representation of the strategy defined in the proof since for all histories $s_1 s_{1m_1} ... s_k$ where player 1 has another action available than $*$ we have

$$\begin{aligned}
&\sigma^1(s_1 s_{1m_1} ... s_k) \\
&= G'(\mathcal{T}'((w_{01}, ..., w_{0N}), s_1 s_{1m_1} ... s_{k-1m_{k-1}}), s_k) \\
&= G'((\mathcal{T}_1(w_{01}, s_1 s_2 ... s_{k-1}), ..., \mathcal{T}_N(w_{0N}, s_1 s_2 ... s_{k-1})), s_k) \\
&= (G_1(\mathcal{T}_1(w_{01}, s_1 s_2 ... s_{k-1}), s_k), ..., G_N(\mathcal{T}_N(w_{0N}, s_1 s_2 ... s_{k-1}), s_k)) \\
&= (\sigma_1(s_1 s_2 ... s_k), ... \sigma_N(s_1 s_2 ... s_k))
\end{aligned}$$

where $\mathcal{T}'$ is the repeated transition function of the Mealy automaton $\mathcal{A}$ and $\mathcal{T}_j$ is the repeated transition function the Mealy automaton $\mathcal{A}_j$ for $1 \leq j \leq N$.

($\Leftarrow$) For the second direction, we assume σ is given by the Mealy automaton $\mathcal{A} = (W, w_0, \text{States}', \text{Act}', T, G)$ where $\text{States}' = \text{States} \cup \{s_m | s \in \text{States}, m \in \text{Act}^N, m_\ell \in \text{Mov}(s, \ell) \text{ for } 1 \leq \ell \leq N\}$ and $\text{Act}' = \text{Act}^N \cup \{*\}$. Now define σ^1 by the Mealy automata $\mathcal{A}_\ell = (W, w_0, \text{States}, \text{Act}, T_\ell, G_\ell)$ for $1 \leq \ell \leq N$ where for all $w \in W$ and $s \in \text{States}$

- $T_\ell(w, s) = T(T(w, s), s_{G(w,s)_\ell})$
- $G_\ell(w, s) = G(w, s)_\ell$

This is a finite-memory representation of the strategy defined in the proof since for all sequences $s_1 s_2 ... s_k \in \text{Hist}_\mathcal{G}$ and all $1 \leq \ell \leq N$

$$\begin{aligned}
&\sigma_\ell(s_1 s_2 ... s_k) \\
&= G_\ell(\mathcal{T}_\ell(w_0, s_1 s_2 ... s_{k-1}), s_k) \\
&= G(\mathcal{T}(w_0, s_1 s_{1m_1} s_2 s_{2m_2} ... s_{k-1} s_{k-1m_{k-1}}), s_k)_\ell \\
&= \sigma^1(s_1 s_{1m_1} s_2 s_{2m_2} ... s_{k-1} s_{k-1m_{k-1}} s_k)_\ell
\end{aligned}$$

where $m_j = G(\mathcal{T}(w_0, s_1 s_{1m_1} ... s_j))$ for all j. $\mathcal{T}$ is the repeated transition function for $\mathcal{A}$ and $\mathcal{T}_\ell$ is the transition function for $\mathcal{A}_\ell$.

This finishes the proof for finite-memory strategies.

Finally, in case 3. the proof is the same as in case 2. However, we need to show that the strategies generated are memoryless, if the strategies from which they are generated are memoryless. This follows directly from the construction of the finite-memory strategies above, since if W_j are singleton sets for $1 \leq j \leq N$, then $\prod_{j=1}^{N} W_j$ is also a singleton set. □

This lemma allows us to find memoryless, finite-memory and perfect recall sure-winning strategies for LTL objectives for coalitions in concurrent games by using results from turn-based games. In [PR89] it is proved that in turn-based games with LTL objectives only finite memory is needed in winning strategies. Together with Lemma 10 this means that for a coalition of players in a concurrent game there is a sure-winning strategy for an LTL objective if and only if there is a sure-winning finite-memory strategy.

Corollary 11 *For all LTL formulae φ, all concurrent game structures $\mathcal{M}$ and all paths ρ we have*

$$\mathcal{M}, \rho \models_{IR} \langle\langle A \rangle\rangle \varphi \text{ if and only if } \mathcal{M}, \rho \models_{IF} \langle\langle A \rangle\rangle \varphi$$

The next lemma tells us that for a fixed iCGM $\mathcal{M}$ and an ATL^* formula φ, an LTL formula φ' can be constructed that is true along the same paths as φ if we extend the labeling of the states in $\mathcal{M}$ with additional proposition symbols. This is done by replacing each state-formula φ_s in φ with a new proposition p_s that is true exatcly in the states where φ_s is true. This is a well-known technique which is also used when doing model-checking of CTL^* [CES86]. Here we show that it can be used in ATL^* as well, even with incomplete information.

Lemma 12 *For a fixed iCGM $\mathcal{M} = (\mathcal{G}, \pi)$, a given ATL^* formula φ over the set* Prop *of propositions, $X \in \{r, F_1, F_2, ..., F, R\}$ and $Y \in \{i, I\}$ there is an LTL formula φ' where $|\varphi'| \leq |\varphi|$ and an extension π' of π such that for all paths ρ in $\mathcal{G}$ we have*

$$(\mathcal{G}, \pi), \rho \models_{YX} \varphi \text{ if and only if } (\mathcal{G}, \pi'), \rho \models_{YX} \varphi'$$

.

Proof. Let $\mathcal{M} = (\mathcal{G}, \pi)$ be a fixed iCGM, let φ be an ATL^* formula over the set Prop of proposition symbols, let $X \in \{r, F_1, F_2, ..., F, R\}$ and $Y \in \{i, I\}$. An

LTL formula φ' with the desired property is obtained by replacing the outermost state subformulae $\varphi_1, ..., \varphi_k$ of φ by new proposition symbols $p_1, ..., p_k \notin \text{Prop}$. The extension π' is defined by $\pi'(s) = \pi(s) \cup \{p_j | \mathcal{M}, s \models_{YX} \varphi_j\}$ for all $s \in$ States. Thus, the proposition p_j is true in a state s whenever φ_j evaluates to true in s with the given type of semantics.

We do the proof by induction on the structure of φ. For the base case, where $\varphi = p$ for some proposition $p \in \text{Prop}$ we have for all paths ρ

$$(\mathcal{G}, \pi), \rho \models_{YX} p \text{ iff } p \in \pi(\rho_0) \text{ iff } p \in \pi'(\rho_0) \text{ iff } (\mathcal{G}, \pi'), \rho \models_{YX} p$$

As induction hypothesis suppose the Lemma is true for all subformulae φ_1 and φ_2 of φ, witnessed by the *LTL* formulae φ_1' and φ_2' and the extended labeling functions π_1' and π_2' (which are not extended by the same proposition symbols, i.e. $\pi_1'(s) \cap \pi_2'(s) = \pi(s)$ for all $s \in$ States) obtained by the same procedure as for φ. We now consider the different cases.

For the cases where φ is a state formula (of the form $\neg\varphi_1, \varphi_1 \vee \varphi_2$ or $\langle\langle A \rangle\rangle \varphi_1$) φ is just replaced by a proposition $p \notin \text{Prop}$ to obtain φ' and $\pi'(s) = \pi(s) \cup \{p | (\mathcal{G}, \pi), s \models_{YX} \varphi\}$ for all $s \in$ States. Then we have for all paths ρ

$$(\mathcal{G}, \pi), \rho \models_{YX} \varphi \text{ iff } (\mathcal{G}, \pi), \rho_0 \models_{YX} \varphi \text{ iff } p \in \pi'(\rho_0) \text{ iff } (\mathcal{G}, \pi'), \rho \models_{YX} p$$

For the cases where φ is a path formula, we have the following

$\varphi = \mathbf{X}\varphi_1$:

For all paths ρ

$$(\mathcal{G}, \pi), \rho \models_{YX} \mathbf{X}\varphi_1 \text{ iff } (\mathcal{G}, \pi), \rho_{\geq 1} \models_{YX} \varphi_1 \text{ iff } (\mathcal{G}, \pi_1'), \rho_{\geq 1} \models_{YX} \varphi_1' \text{ iff}$$

$$(\mathcal{G}, \pi_1'), \rho \models_{YX} \mathbf{X}\varphi_1'$$

by using the induction hypothesis. Here $\mathbf{X}\varphi_1'$ is an *LTL* formula since φ_1' is an LTL formula and π_1' extends π.

$\varphi = \varphi_1 \mathbf{U} \varphi_2$:

For all paths ρ

$$(\mathcal{G}, \pi), \rho \models_{YX} \varphi_1 \mathbf{U} \varphi_2 \text{ iff}$$

$$\exists k. (\mathcal{G}, \pi), \rho_{\geq k} \models_{YX} \varphi_2 \text{ and } \forall j < k. (\mathcal{G}, \pi), \rho_{\geq j} \models_{YX} \varphi_1 \text{ iff}$$

$\exists k.(\mathcal{G}, \pi_2'), \rho_{\geq k} \models_{YX} \varphi_2'$ and $\forall j < k.(\mathcal{G}, \pi_1'), \rho_{\geq j} \models_{YX} \varphi_1'$ iff

$\exists k.(\mathcal{G}, \pi_1' \cup \pi_2'), \rho_{\geq k} \models_{YX} \varphi_2'$ and $\forall j < k.(\mathcal{G}, \pi_1' \cup \pi_2'), \rho_{\geq j} \models_{YX} \varphi_1'$ iff

$(\mathcal{G}, \pi_1' \cup \pi_2'), \rho \models_{YX} \varphi_1' \mathbf{U} \varphi_2'$

by using the induction hypothesis. Here $\varphi_1' \mathbf{U} \varphi_2'$ is an LTL formula since φ_1' and φ_2' are LTL formulae and $\pi_1' \cup \pi_2'$ extends π. □

The idea of replacing the outermost state subformulae of a formula with new propositions used in the proof above also gives us the following for ATL

Corollary 13 *For a fixed iCGM $\mathcal{M} = (\mathcal{G}, \pi)$, a given ATL formula φ over the set* Prop *of propositions, $X \in \{r, F_1, F_2, ..., F, R\}$ and $Y \in \{i, I\}$ there exists an extension π' of π with image* Prop$\cup\{p\}$ *for some proposition symbol $p \notin$* Prop *such that for all paths ρ in $\mathcal{G}$ we have*

$$(\mathcal{G}, \pi), \rho \models_{YX} \varphi \text{ if and only if } (\mathcal{G}, \pi'), \rho \models_{YX} p$$

Note that Lemma 12 and Corollary 13 are declarative in the sense that they do not provide algorithms for actually computing the extensions of the labeling functions. This is not too important for the analysis of expressiveness we are doing in this chapter, however it is quite essential for the model-checking problem. To find the states in which a state formula is true all state subformulae are evaluated first, starting with the innermost subformula. In ATL^* this leads to the problem of finding the states in which a formula of the form $\langle\langle A \rangle\rangle \varphi$ is true, where φ is an LTL formula. For ATL this leads to evaluating formulae of the form $\langle\langle A \rangle\rangle \mathbf{G} \varphi_1$, $\langle\langle A \rangle\rangle \varphi_1 \mathbf{U} \varphi_2$ and $\langle\langle A \rangle\rangle \mathbf{X} \varphi_1$ where φ_1 and φ_2 are formulae of classical propositional logic. Using Lemma 10 this leads to turn-based games with essentially the same type of objectives and gives us an intuition about why we need finite memory in ATL^* and not in ATL. This is because games with LTL objectives require finite memory whereas the games corresponding to $\langle\langle A \rangle\rangle \mathbf{G} \varphi_1$ in ATL are safety games, the games corresponding to $\langle\langle A \rangle\rangle \varphi_1 \mathbf{U} \varphi_2$ are reachability games and the games corresponding to $\langle\langle A \rangle\rangle \mathbf{X} \varphi_1$ are two-step reachability games (by substituting $\mathbf{X}$ with $\mathbf{XX}$ when generating the turn-based game). These classes of games do not require memory for winning strategies [EJ91]. These ideas lead to the following results.

Proposition 14 *For all ATL formulae φ, all concurrent game structures $\mathcal{M}$ and all paths ρ we have*

$$\mathcal{M}, \rho \models_{IR} \varphi \text{ if and only if } \mathcal{M}, \rho \models_{Ir} \varphi$$

Since $\mathbb{M}_{Ir}^{ATL}$ is trivially contained in $\mathbb{M}_{IX}^{ATL}$ for $X \in \{F, R, F_2, F_3, ...\}$ all the different types of semantics introduced are actually the same in the case of ATL

Corollary 15 *For all ATL formulae φ, all concurrent game structures $\mathcal{M}$ and all paths ρ we have*

$$\mathbb{M}_{Ir}^{ATL} = \mathbb{M}_{IF_2}^{ATL} = \mathbb{M}_{IF_3}^{ATL} = ... = \mathbb{M}_{IF}^{ATL} = \mathbb{M}_{IR}^{ATL}$$

We proceed with the following result stating that $\mathbb{M}_{IR}^{ATL^*} = \mathbb{M}_{IF}^{ATL^*}$. This means that we only have to consider finite-memory strategies in perfect information games with ATL^*.

Theorem 16 *For all ATL* formulae φ, all concurrent game structures $\mathcal{M}$ and all paths ρ we have*

$$\mathcal{M}, \rho \models_{IR} \varphi \text{ if and only if } \mathcal{M}, \rho \models_{IF} \varphi$$

Proof. The proof is done by induction on the structure of φ. First, we have the base case

$\varphi = p$:

For all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ we have

$$\mathcal{M}, \rho \models_{IR} p \text{ if and only if } p \in \pi(\rho_0) \text{ if and only if } \mathcal{M}, s \models_{IF} p$$

We now assume that the theorem is true for all subformulae of φ. The induction cases are as follows

$\varphi = \varphi_1 \vee \varphi_2$:

For all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ we have

$\mathcal{M}, \rho \models_{IR} \varphi_1 \vee \varphi_2$ if and only if

$\mathcal{M}, \rho \models_{IR} \varphi_1$ or $\mathcal{M}, \rho \models_{IR} \varphi_2$ if and only if

$\mathcal{M}, \rho \models_{IF} \varphi_1$ or $\mathcal{M}, \rho \models_{IF} \varphi_2$ if and only if

$\mathcal{M}, \rho \models_{IF} \varphi_1 \vee \varphi_2$

by using the induction hypothesis.

$\varphi = \neg\varphi_1$:

For all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ we have

$\mathcal{M}, \rho \models_{IR} \neg\varphi_1$ if and only if

$\mathcal{M}, \rho \not\models_{IR} \varphi_1$ if and only if

$\mathcal{M}, \rho \not\models_{IF} \varphi_1$ if and only if

$\mathcal{M}, \rho \models_{IF} \neg\varphi_1$

by using the induction hypothesis.

$\varphi = \mathbf{X}\varphi_1$:

For all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ we have

$\mathcal{M}, \rho \models_{IR} \mathbf{X}\varphi_1$ if and only if

$\mathcal{M}, \rho_{\geq 1} \models_{IR} \varphi_1$ if and only if

$\mathcal{M}, \rho_{\geq 1} \models_{IF} \varphi_1$ if and only if

$\mathcal{M}, \rho \models_{IF} \mathbf{X}\varphi_1$

by using the induction hypothesis.

$\varphi = \varphi_1 \mathbf{U} \varphi_2$:

For all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ we have

$\mathcal{M}, \rho \models_{IR} \varphi_1 \mathbf{U} \varphi_2$ if and only if

$\exists k. \mathcal{M}, \rho_{\geq k} \models_{IR} \varphi_2$ and $\forall j < k. \rho_{\geq j} \models_{IR} \varphi_2$ if and only if

$\exists k. \mathcal{M}, \rho_{\geq k} \models_{IF} \varphi_2$ and $\forall j < k. \rho_{\geq j} \models_{IF} \varphi_2$ if and only if

$$\mathcal{M}, \rho \models_{IF} \varphi_1 \mathbf{U} \varphi_2$$

by using the induction hypothesis.

$\varphi = \langle\langle A \rangle\rangle \varphi_1$:

For the first direction we have for all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ

$$\mathcal{M}, \rho \models_{IF} \langle\langle A \rangle\rangle \varphi_1 \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A^F . \forall \rho' \in \mathrm{Out}(\rho_0, \sigma_A) . \mathcal{M}, \rho' \models_{IF} \varphi_1$$

By the induction hypothesis this implies

$$\exists \sigma_A \in \mathrm{Strat}_A^F . \forall \rho' \in \mathrm{Out}(\rho_0, \sigma_A) . \mathcal{M}, \rho' \models_{IR} \varphi_1 \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A . \forall \rho' \in \mathrm{Out}(\rho_0, \sigma_A) . \mathcal{M}, \rho' \models_{IR} \varphi_1 \text{ implies}$$

$$\mathcal{M}, \rho \models_{IR} \langle\langle A \rangle\rangle \varphi_1$$

For the other direction we use Lemma 12 which says that there is an LTL formula φ_1^{LTL} and an extension π' of π such that for all paths ρ we have $(\mathcal{G}, \pi), \rho \models_{IR} \varphi_1$ if and only if $(\mathcal{G}, \pi'), \rho \models_{IR} \varphi_1^{LTL}$. Denote $\mathcal{M}' = (\mathcal{G}, \pi')$. Using this, we have for all CGMs $\mathcal{M} = (\mathcal{G}, \pi)$ and paths ρ

$$\mathcal{M}, \rho \models_{IR} \langle\langle A \rangle\rangle \varphi_1 \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A . \mathcal{M}, \rho' \models_{IR} \varphi_1 \text{ for every } \rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A) \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A . \mathcal{M}', \rho' \models_{IR} \varphi_1^{LTL} \text{ for every } \rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A) \text{ implies}$$

$$\mathcal{M}', \rho \models_{IR} \langle\langle A \rangle\rangle \varphi_1^{LTL}$$

By Corollary 11 this implies

$$\mathcal{M}', \rho \models_{IF} \langle\langle A \rangle\rangle \varphi_1^{LTL} \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A^F . \mathcal{M}', \rho' \models_{IF} \varphi_1^{LTL} \text{ for every } \rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A) \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A^F . \mathcal{M}', \rho' \models_{IR} \varphi_1^{LTL} \text{ for every } \rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A) \text{ implies}$$

$$\exists \sigma_A \in \mathrm{Strat}_A^F . \mathcal{M}, \rho' \models_{IR} \varphi_1 \text{ for every } \rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A)$$

By the induction hypothesis this implies

$\exists \sigma_A \in \mathrm{Strat}_A^F.\mathcal{M}, \rho' \models_{IF} \varphi_1$ for every $\rho' \in \mathrm{Out}_{\mathcal{G}}(\rho_0, \sigma_A)$ implies

$\mathcal{M}, \rho \models_{IF} \langle\langle A \rangle\rangle \varphi_1$

which finishes the last case. □

We have used well-known results from turn-based games in order to prove that only memoryless strategies are needed for ATL and only finite-memory strategies are needed for ATL^* with complete information. In addition, the insights gained about the underlying turn-based games and their winning objectives provide us with intuition about why finite memory is needed for ATL^* and not for ATL. The technique used in this chapter, however, does not work for incomplete information games. The reason lies in the difference of information available to the different players. For complete information there is no problem in letting one player (player 1 in the turn-based game) control the whole coalition A, since all players in A have the same information available at all times. However, in an incomplete information game, player 1 cannot have the same information available when acting for the different players in A, because then he can choose from strategies that are not available to the players in A due to their lack of information. Another reason supporting that the technique does not work for incomplete information is to be found in the undecidability of model-checking ATL with incomplete information (even with as simple a formula as $\langle\langle A \rangle\rangle \mathrm{G} p$ [AHK02, DT11]) and the fact that finding winning strategies in two-player turn-based games with incomplete information and ω-regular objectives (and by implication with LTL objectives) is decidable [RCDH07].

Chapter 5

ATL/ATL^* Model-Checking

5.1 Overview

Model-checking is a technique that is traditionally used in computer science to do formal verification of systems. The idea is to take a formal model M_S of a system S and a formula φ_R corresponding to a requirement specification R of the system and checking whether the formal model M_S indeed satisfies the formal specification φ_R. This technique is typically used to verify that a given system satisfies some property by mathematically proving that the model of the system satisfies the formula which represents the desired behavior. This means that both the modeling of the system and the formalization of the requirement specification are very important because errors in the modeling phase might hide undesired behavior of the system. Since one cannot be completely sure that modeling is done without any errors, model-checking is typically used as a complementary technique to testing. The linear-time temporal logic LTL and the branching-time temporal logics CTL/CTL^* have been the most popular logical languages for formalizing the requirement specifications of programs in this context, see e.g. [CGP01, BK08]. In this thesis we have motivated the use of alternating-time temporal logics for specifying properties of concurrent programs and distributed systems. A natural extension is to examine the possibility of using this form of specification in the context of model-checking. Since ATL^* includes the logics LTL, CTL and CTL^*, a model-checker for ATL^* has

at least the same capabilities as a model-checker for any of the other three logical languages. In addition to using model-checking of ATL/ATL^* for verification of systems, we can also use it to do synthesis of programs. This can be done since concurrent game models can be used to describe capabilities of different agents in an environment. Kripke structures that are usually used for model-checking LTL, CTL and CTL^* are only used to describe systems that have already been implemented, in which non-deterministic behavior makes different executions of a system possible. Consider the process of model-checking an ATL/ATL^* formula $\langle\langle A\rangle\rangle\varphi$ in a concurrent game model $\mathcal{M}$, where $\mathcal{M}$ is a model of the interaction between devices in some environment, the agents in coalition A correspond to the subset of the devices we wish to program and φ is a formal specification of the behavior that we would like any execution to exhibit. What we are actually doing is to ask whether there exist programs for the devices represented by A that make sure the formal specification φ is satisfied. The model-checking algorithms we present in this chapter are all constructive, which means that when $\langle\langle A\rangle\rangle\varphi$ is model-checked, then a witness-strategy for coalition A is given as output that makes sure that φ is satisfied. Such a strategy can be converted into actual programs for the devices that are represented by the agents in A, and these will provably satisfy the formal specification, given that the modeling has been done precisely. Thus, ATL/ATL^* model-checking can in fact be used for generating programs with provably correct properties with respect to a model. However, the capability of generating programs automatically from formal specifications comes with a computational price. In this chapter we will not be concerned with the modeling of systems and requirement specifications, but simply focus on complexity and algorithms of the core model-checking problem: Given an (i)CGM $\mathcal{M}$, an initial state s, an ATL/ATL^* formula φ and a given type of semantics $\models_{XY}$ does $\mathcal{M}, s \models_{XY} \varphi$?

When analyzing the model-checking complexity of the different semantics we assume that the number of players in a concurrent game structure is fixed. Let the input to the model-checking problem consist of an iCGM $\mathcal{M} = (\mathcal{G}, \pi)$ over a finite set Prop of proposition symbols, a state s and an ATL/ATL^* formula φ, where $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$. Then the input size is the sum of the number of states in $\mathcal{G}$, the number of transitions in $\mathcal{G}$, the size of the labeling function π represented explicitly and the size of the formula φ. Thus, the size of the input is $|\text{States}|+|\text{Tab}|+|\text{Prop}|\cdot|\text{States}|+|\varphi|$. This approach is used in most sources in the litterature, but other works have been done where the number of agents is a parameter, and where the number of transitions is assumed to increase exponentially in the number of players [JD08, Jam08, BDJ10]. So the reader should keep in mind that the complexity results presented here are with a fixed number of players and use the game structure as input, in which the number of agents is implicitly included. When model-checking with bounded-memory semantics with a memory size bounded by k we also use the memory size as input.

	ATL	ATL^*
$\models_{Ir}$	$PTIME$ [AHK02]	$PSPACE$ [Sch04]
$\models_{IF_k}$	$PTIME$	$PSPACE$
$\models_{IF}$	$PTIME$	$2EXPTIME$
$\models_{IR}$	$PTIME$ [AHK02]	$2EXPTIME$ [AHK02]

	ATL	ATL^*
$\models_{ir}$	Δ_2^p [Sch04, JD08]	$PSPACE$ [Sch04]
$\models_{iF_k}$	Δ_2^p	$PSPACE$
$\models_{iF}$	?	Undecidable
$\models_{iR}$	Undecidable [AHK02, DT11]	Undecidable [AHK02, DT11]

Figure 5.1: Model-checking complexity for ATL and ATL^*. All the complexity results are completeness results

The complexity of model-checking with the different types of semantics are presented in Figure 5.1, where all the complexity results shown are completeness results. The results for $\models_{ir}, \models_{iR}, \models_{Ir}$ and $\models_{IR}$ semantics have already been obtained in the litterature and the results in the other cases will be proven in the rest of this chapter. Some of the new results are positive whereas others are negative. A negative result is that finite-memory semantics is as hard as perfect recall semantics in all the cases except ATL with incomplete information where the decidability of the model-checking problem is still open. This means that we still have undecidability in ATL^* with incomplete information and still have $2EXPTIME$-completeness with complete information for ATL^*. Even though the complexity results are the same we will show in this chapter that there exists incomplete information games where $\models_{iR}$ and $\models_{iF}$ semantics are different by showing that infinite memory can be necessary to win a game for a coalition of players. As expected we still have $PTIME$-completeness for ATL in the complete information case which is of course positive. For bounded memory semantics we get the same complexity results as for memoryless semantics in all cases which is very positive. There is an increase in running time of the algorithms when going from memoryless to finite-memory strategies with a bounded size, but the increase will be shown not to be too large in any of the cases. Especially it is a positive result that even though model-checking ATL^* with memoryless strategies and complete information is already $PSPACE$-complete, we still stay in the same complexity class when we add incomplete information to the model and consider bounded-memory strategies. This is a nice result, since in all cases it can be shown that adding incomplete information always makes the problem harder, and increasing the memory allowed also makes the problem harder. It is also interesting to see that adding incomplete information and bounded-

memory strategies do not make the problem much harder, whereas going from memoryless strategies and complete information to perfect recalll strategies and complete information in the case of ATL^* takes the problem from $PSPACE$ to $2EXPTIME$-complete. Such results are interesting since in all applications where infinite memory is not needed and the amount of memory available is known, the algorithms will be significantly faster. Especially when considering the application of model-checking in program synthesis bounded-memory seems to provide a good model. This is because infinite memory is not available in practice and the specification of the hardware on which the programs are to run is typically known, which gives a bound on the memory size. Note that model-checking ATL with incomplete information and memoryless/bounded-memory semantics is complete for the class $\Delta_2^p = P^{NP}$ in the polynomial hierarchy. This class contains NP and is contained in $PSPACE$ and is defined in Appendix C.

5.2 Memoryless and perfect recall semantics

The results for memoryless and perfect recall semantics were obtained in [AHK02, Sch04, JD08, DT11] as shown in Figure 5.1. In this section we superficially explain how the algorithms work in the decidable cases. The algorithms can, together with our expressiveness results from the previous chapter, be altered to handle the cases of bounded-memory semantics as well as finite-memory semantics.

For a proof of the undecidability result in the case of perfect recall and incomplete information we refer to [DT11] in which model-checking of the ATL formula $\langle\langle\{1,2\}\rangle\rangle\mathbf{G}p$ for a proposition symbol p is shown undecidable in 3-player game structures. This is done by a reduction from the non-halting problem for Turing machines. Later in this chapter we will present a new proof that ATL^* model-checking is undecidable, both with perfect recall and finite-memory semantics. This is done since the proof in [DT11] requires infinite memory strategies and it does not seem like it can be easily altered to handle the finite-memory case. However, the proof we present only works for ATL^* and we leave it as an open question if model-checking ATL with finite-memory semantics and incomplete information is decidable or not.

In all the decidable cases, the algorithms are inspired by model-checking algorithms for CTL and CTL^* and algorithms for sure-winning in turn-based games. We postpone the explanation of the $PSPACE$-algorithm of ATL^* with memoryless semantics to the next section, since it is contained in the bounded-memory case. This is done for the Δ_2^p-algorithm for ATL with memoryless semantics and incomplete information as well.

We first focus on the $PTIME$ complexity for ATL with complete information. Because of Corollary 15 only memoryless strategies are needed and therefore $\models_{Ir}$, $\models_{IF_k}$, $\models_{IF}$ and $\models_{IR}$ are equivalent for all k for ATL with complete information. We can therefore treat all these cases at the same time. We will use the technique from Section 4.3 of generating a two-player turn-based game $\mathcal{M}_A$ from a given concurrent game $\mathcal{M}$ and a coalition A of players in $\mathcal{M}$ such that the coalition A has a strategy to make sure that φ is satisfied in $\mathcal{M}$ if and only if player 1 has a sure-winning strategy for the objective $\varphi[\mathbf{X} \mapsto \mathbf{XX}]$ in $\mathcal{M}_A$. We first need a few lemmas which state that we can solve two-player turn-based reachability and safety games effectively. These facts have been known for a long time, for an overview see e.g. [dAH00].

Lemma 17 *The winning states of a two-player turn-based reachability game can be found in PTIME.*

Proof. Suppose we are given a two-player turn-based CGM $\mathcal{M} = (\mathcal{G}, \pi)$ where $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab})$ with an objective $\mathbf{F}p$ for some proposition symbol p for player 1. The algorithm for finding winning states works as a fixpoint computation, working backwards from the reachability set $R = \{s \in \text{States} | p \in \pi(s)\}$. Let

$$\text{Pre}(X)\{s \in \text{States} | \exists a_1 \in \text{Mov}(s,1).\forall a_2 \in \text{Mov}(s,2).\text{Tab}(s,(a_1,a_2)) \in X\}$$

for a given subset X of states. This denotes the set of states in which player 1 can be sure to reach the set X in 1 step. We now define a sequence W_j of states for $j \geq 0$ as follows

$$W_0 = R$$

$$W_{j+1} = W_j \cup \text{Pre}(W_j)$$

This sequence is increasing, and since the set of states is finite the sequence converges. Let W denote the limit of the sequence, and let c be the least index such that $W_c = W$. We will show that W is the set of winning states for player 1, i.e. the set of states from which he has a sure-winning strategy. Since the sequence is strictly increasing until it converges, we have to calculate W_j for at most |States| different values of j and each calculation can be done in time polynomial in the game structure.

The sets W_j can be thought of the sets from which player 1 can be sure to reach R in at most j steps of the game. Let $f : \text{States} \to \mathbb{N} \cup \{\infty\}$ be a function

of states such that $f(s)$ is the least index such that $s \in W_{f(s)}$ for $s \in W$ and $f(s) = \infty$ otherwise. Thus, $f(s)$ is the step of the computation in which s was added to W.

First we will show that if a state $s \in W$, then s is winning for player 1. Suppose $s \in W$ and $f(s) = i$ for some $i > 0$ (if f(s) = 0, then player 1 is trivially winning). Then there exists an action a_1 such that for any action a_2 of player 2 we have $\mathrm{Tab}(s, (a_1, a_2)) = s'$ where $f(s') < f(s)$. We design a strategy σ for player 1 that chooses such an action a_1 in s. We design σ in the same way for all states $t \in W$ with $f(t) > 0$. If a given play $\rho = s_1 s_2 ...$, where player 1 plays according to σ starts in state $s = s_1 \in W$ then $f(s_1)f(s_2)...$ is a decreasing sequence until at some point $f(s_j) = 0$ for some j. Since $f(s_j) = 0$ if and only if $s_j \in R$ then σ is a winning strategy from s.

Now suppose that $s \notin W$. Then for any action a_1 for player 1 in s, there exists a counter-action a_2 for player 2 such that $\mathrm{Tab}(s, (a_1, a_2)) = s'$ where $s' \notin W$. For a given strategy σ_1 of player 1, we can define a counter-strategy σ_2 for player 2 choosing such actions in all states $t \notin W$. Then $\mathrm{Out}(s, (\sigma_1, \sigma_2)) = s_1 s_2 ...$ will have $s_j \notin W$ for all $j \geq 0$ and therefore player 1 is not sure winning. □

Note that the winning set for two-player turn-based reachability games can also be expressed by the μ-calculus formula $\mu X.(R \vee \mathrm{Pre}(X))$ as a least fixpoint. Two-player turn-based games are determined with the general class of Borel objectives [Mar75], which include safety and reachability objectives. Thus, player 1 is sure-winning from a state s with reachability objective $\mathbf{F}p$ if and only if player 2 is not sure-winning from s with the complementary objective $\neg\mathbf{F}p$. This means that the winning states of player 2 with the safety objective $\neg\mathbf{F}p$ is simply calculated as the comlement of the winning states of player 1 with the reachability objective $\mathbf{F}p$. Thus,

Lemma 18 *The winning states of a two-player turn-based safety game can be found in PTIME.*

If player 1 is playing a safety game trying to avoid the set R of states, then we can express the winning sets by the μ-calculus formula $\nu X.(\neg R \wedge \mathrm{Pre}(X))$. We are now ready to present the $PTIME$ algorithm for model-checking ATL with complete information.

Proposition 19 *Model-checking ATL with $\models_{Ir}$ semantics is PTIME-complete.*

Proof. As in CTL model-checking, we perform model-checking of an ATL formula φ with complete information in a CGM $\mathcal{M} = (\mathcal{G}, \pi)$ by model-checking

state subformulae, starting with the innermost subformula and moving outwards. Every time a subformula φ_1 has been processed, the states in which it is true is labeled by a new proposition p_1 representing the subformula and the occurence of φ_1 in φ is replaced by p_1 as well. According to Corollary 13 this does not change the set of paths on which φ are true. This procedure continues until we have finally model-checked φ itself, as it is the outermost state subformula. In the cases where the main connectives of a subformula being processed are boolean connectives, we simply perform set complementation in the case of negation and set union in the case of disjunction. This leaves the three cases $\langle\langle A\rangle\rangle \mathbf{X}p$, $\langle\langle A\rangle\rangle \mathbf{G}p$ and $\langle\langle A\rangle\rangle (p\mathbf{U}q)$ for propositions p and q. In these three cases we use our reduction to turn-based games and generate the two-player turn-based game $\mathcal{M}_A = (\mathcal{G}_A, \pi_A)$, which is linear in the size of $\mathcal{M}$ and where coalition A is winning from state s in $\mathcal{M}$ with LTL objective φ if and only if player 1 is winning from s in $\mathcal{M}_\mathcal{A}$ with objective $\varphi[\mathbf{X} \mapsto \mathbf{XX}]$ according to Lemma 10. Then we do as follows

- $\langle\langle A\rangle\rangle \mathbf{G}p$. In this case the game is a safety game for player 1 which can be solved in $PTIME$ according to Lemma 18.
- $\langle\langle A\rangle\rangle (p\mathbf{U}q)$. In this case we remove all outgoing transitions from states in $\mathcal{M}_A$ where both p and q are false and add self-loops to these states. The game is then solved as a reachability game with reachability objective $\mathbf{F}q$ which can be done in $PTIME$ according to Lemma 17.
- $\langle\langle A\rangle\rangle \mathbf{X}p$. In this case the objective in the turn-based game becomes $\mathbf{XX}p$ for player 1. Using the Pre operator used in the proof of Lemma 17, we can find the set of winning states for player 1 as $\mathrm{Pre}(\mathrm{Pre}(R))$ where $R = \{s \in \mathrm{States}_{M_A} | p \in \pi_A(s)\}$. This can be calculated in $PTIME$.

Thus, the algorithm runs in $PTIME$.

$PTIME$-hardness can be proved by a reduction from the CircuitValue problem with only AND and OR gates which is $PTIME$-complete [AB09]. Every instance of this problem can be reduced to a two-player turn-based reachability game in which player 1 controls the OR-gates and player 2 controls the AND-gates. The play then starts at the output state and player 1 wins if a positive input is reached at the end and player 2 wins otherwise. Then the circuit evaluates to true, if and only if player 1 has a strategy that makes sure a positive input state in the corresponding turn-based game is reached. Since checking existence of a sure-winning strategy from a state s with a reachability objective is the same as model-checking the ATL formula $\langle\langle\{1\}\rangle\rangle \mathbf{F}p$ this completes the proof. □

For model-checking ATL^* with perfect recall semantics we apply the following lemma, which is proved in [PR89].

Lemma 20 *For a two-player turn-based game with LTL objectives, finding the set of winning states can be done in* $2EXPTIME$.

Now we are ready to show the following

Proposition 21 *Model-checking* ATL^* *with* $\models_{IR}$ *and complete information is* $2EXPTIME$*-complete.*

Proof. Just as the $PTIME$ algorithm for ATL resembles model-checking of CTL, the $2EXPTIME$ algorithm for ATL^* resembles model-checking of CTL^*. We use the same trick of processing state subformulae one at a time, starting with the innermost and moving outwards which is possible according to Lemma 12. When processing formulae where the main connective is a boolean connective, we do as in the ATL case. When processing formulae where the main connective is a strategy quantifier $\langle\langle A\rangle\rangle\varphi$ for an LTL formula φ there is a difference though. First, we do as in the case of ATL model-checking and construct the two-player turn-based game $\mathcal{M}_A$ and the objective $\varphi[\mathbf{X} \mapsto \mathbf{XX}]$ for player 1. Then, we find the set of states from which player 1 has a sure-winning strategy which can be done in $2EXPTIME$ according to Lemma 20. Since there are at most a linear number (in the size of the formula) of calls to this $2EXPTIME$ procedure, the whole algorithm runs in $2EXPTIME$.

The 2EXPTIME hardness is proven in [AHK02] as a reduction from the realizability problem for LTL [PR89]. □

5.3 Bounded-memory semantics

In this section we consider bounded-memory semantics. Results from the previous section can be reused in some cases. In other cases we present algorithms that work by non-deterministically guessing bounded-memory strategies of a given size and verifying that the guessed strategies satisfy some property. This technique is possible because of the bound on the memory-size and therefore cannot in general be reused for finite-memory and perfect recall semantics. We start by stating the following result

Proposition 22 *Model-checking ATL with $\models_{IF_k}$ semantics is PTIME-complete*

Proof. Since only memoryless strategies are needed in ATL with complete information according to Corollary 15, we have that $M, s \models_{Ir} \varphi$ iff $M, s \models_{IF_k} \varphi$ for all ATL formulae φ and all $k \geq 1$. □

We have the following relationships between some of the model-checking problems, giving us a better indication about the hardness of some of the problems considered.

Proposition 23 *For any $X \in \{r, F_k, F, R\}$ and $Y \in \{i, I\}$ we have for both ATL and ATL^* that*

1. *Model-checking with $\models_{iX}$ semantics is at least as hard as $\models_{IX}$ semantics.*
2. *Model-checking with $\models_{YF_k}$ semantics is at least as hard as $\models_{Yr}$ semantics.*

Proof. 1. follows since complete information games are contained in incomplete information games. 2. follows since $\models_{Yr}$ is equivalent to $\models_{YF_1}$ for $Y \in \{i, I\}$ in both ATL and ATL^* the result follows. □

In order to prove Δ_2^p-completeness of model-checking ATL with $\models_{iF_k}$ semantics we first need a lemma which states that model-checking ATL_1 with $\models_{iF_k}$ semantics is NP-complete, as it is with $\models_{ir}$ semantics [Sch04]. Recall that ATL_1 is the fragment of ATL with no nesting of strategic quantifiers. The algorithm in the proof is also the main ingredient in the rest of the algorithms that are presented in this section.

Lemma 24 *ATL_1 is NP-complete with $\models_{iF_k}$ semantics*

Proof. Since model-checking ATL_1 is NP-complete with $\models_{ir}$ semantics [Sch04] we also have NP-hardness with $\models_{iF_k}$ semantics according to Proposition 23

Our NP-algorithm for model-checking an ATL_1 formula $\varphi = \langle\langle A \rangle\rangle \varphi_1$ in a concurrent game model $\mathcal{M} = (\mathcal{G}, \pi)$ with $\mathcal{G} = (\text{States}, \text{Agt}, \text{Act}, \text{Mov}, \text{Tab}, (\sim_j)_{1 \leq j \leq n})$ works by non-deterministically guessing a k-memory strategy $\sigma = (\sigma_j)_{j \in A}$ for coalition A given by Mealy automata $M_j = (Q_j, q_{j0}, []_j, Act, T_j, G_j)$ for $j \in A$ (note that there are only finitely many such strategies) and verifying that the guess satisfies the formula. Because we have to guess at most $|\text{Agt}|$

Mealy automata, each of size polynomial in the input, this guess can be done in polynomial time. We assume without loss of generality that the agents in A are the first m players, i.e. $A = \{1, ..., m\}$ where $m \leq n$. The verification is done by creating a transition system

$$TS_\sigma = (S, R, L)$$

where

- $S = \text{States} \times \prod_{j=1}^{m} Q_j$ is the set of states
- $R \subseteq S \times S$ is a transition relation such that
  $([s, (q_1, ..., q_m)], [s', (q'_1, ..., q'_m)]) \in R$
  if and only if there exists $a_{m+1}, ..., a_n \in \text{Act}$ such that
  - $\text{Tab}(s, (G_1(q_1, [s]_1), ..., G_m(q_m, [s]_m), a_{m+1}, ..., a_n)) = s'$ and
  - $T_j(q_j, [s]_j) = q'_j$ for all $1 \leq j \leq m$
- $L(s, (q_1, ..., q_m)) = \pi(s)$ for all $(q_1, ..., q_m) \in \prod_{j=1}^{m} Q_j$ and $s \in \text{States}$.

The intuition is that the transition system TS_σ represents all the possibilities of agents in $\text{Agt} \setminus A$ given that the players in A play according to σ. This is why we need to save the memory states of the players playing according to σ in the states of TS_σ. We claim that

$\rho = s_0 s_1 s_2 ... \in \text{Out}_{\mathcal{G}}(s_0, \sigma)$ if and only if

$\exists (q'_{1\ell}, ..., q'_{m\ell}) \in \prod_{j=1}^{m} Q_j$ for $\ell \geq 0$ with $q'_{j0} = q_{j0}$ for $1 \leq j \leq m$ so

$(s_1, (q'_{10}, ..., q'_{m0}))(s_2, (q'_{11}, ..., q'_{m1}))...$ is a path in $\mathcal{T}_\sigma$

To see this, we have for all paths ρ

$\rho = s_0 s_1 s_2 ... \in \text{Out}_{\mathcal{G}}(s_0, \sigma)$

if and only if

for all $\ell \geq 0$ there exists $a_{(m+1)\ell}, ..., a_{n\ell} \in \text{Act}$ such that

$\text{Tab}(s_\ell, (\sigma_1(\rho_{\leq \ell}), ..., \sigma_m(\rho_{\leq \ell}), a_{(m+1)\ell}, ..., a_{n\ell})) = s_{\ell+1}$

if and only if

for all $\ell \geq 0$ there exists $a_{(m+1)\ell}, ..., a_{n\ell} \in \mathrm{Act}$ such that

$$\mathrm{Tab}(s_\ell, (G_1(\mathcal{T}_1(q_{10}, \rho_{\leq \ell-1}), [s_\ell]_1), ..., G_m(\mathcal{T}_m(q_{m0}, \rho_{\leq \ell-1}), [s_\ell]_m), a_{(m+1)\ell}, ..., a_{n\ell}))$$

$$= s_{\ell+1}$$

if and only if

for all $\ell \geq 0$ there exists $a_{(m+1)\ell}, ..., a_{n\ell} \in \mathrm{Act}$ such that

$$([s_\ell, (\mathcal{T}_1(q_{10}, \rho_{\leq \ell-1}), ..., \mathcal{T}_m(q_{m0}, \rho_{\leq \ell-1}))],$$

$$[s_{\ell+1}, (\mathcal{T}_1(q_{10}, \rho_{\leq \ell}), ..., \mathcal{T}_m(q_{m0}, \rho_{\leq \ell}))]) \in R$$

if and only if

$\exists (q'_{1\ell}, ..., q'_{m\ell}) \in \prod_{j=1}^{m} Q_j$ for $\ell \geq 0$ with $q'_{j0} = q_{j0}$ for $1 \leq j \leq m$ so

$(s_1, (q'_{10}, ..., q'_{m0}))(s_2, (q'_{11}, ..., q'_{m1}))...$ is a path in TS_σ

In other words, the paths in TS_σ from states with initial memory of all players, correspond exactly to the σ-outcomes in $\mathcal{G}$ which means that

$\mathcal{M}, s \models_{iF_k} \langle\langle A \rangle\rangle \varphi_1$

if and only if

$TS_\sigma, (s', (q_{10}, ..., q_{m0})) \models_{CTL} \mathrm{A}\varphi_1$ for all $s' \in \bigcup_{j \in A} [s]_j$ for some k-bounded strategy $\sigma \in \prod_{j \in A} \mathrm{Strat}_j^{F_k}$ for players in A.

The idea is now that model-checking CTL can be done effectively for a given strategy for players in A and thus after guessing σ non-deterministically, we just need to do CTL model-checking for all the states that are indistinguishable from s for some player in A to determine whether $\mathcal{M}, s \models_{iF_k} \langle\langle A \rangle\rangle \varphi_1$.

The complexity of this algorithm depends on the size of the transition system TS_σ which is as follows:

$|S| = |\mathrm{States}| \cdot k^m \leq k^{|\mathrm{Agt}|}$

$|R| \leq |S| \cdot (|S| - 1)$

which gives a CTL model-checking complexity of $O(|\mathrm{A}\varphi_1| \cdot (k^{|\mathrm{Agt}|} + k^{|\mathrm{Agt}|} \cdot (k^{|\mathrm{Agt}|} - 1))) = O(|\varphi_1| \cdot k^{2|\mathrm{Agt}|})$ for each of the indistinguishable states and thus $O(|\mathrm{States}| \cdot |\varphi_1| \cdot k^{2|\mathrm{Agt}|})$ in total. Now, since the number of agents in a concurrent game model is fixed, this means that we can actually do the check of σ in polynomial time, which means the problem is in NP. □

We are now ready to present the following result, stating that when we use bounded-memory semantics instead of memoryless semantics, we still stay in the complexity class Δ_2^p when model-checking ATL with incomplete information.

Theorem 25 *Model-checking ATL with $\models_{iF_k}$ semantics is Δ_2^p-complete.*

Proof. The ability to model-check ATL_1 formulae with $\models_{iF_k}$ semantics in non-deterministic polynomial time gives us a Δ_2^p algorithm for model-checking ATL with $\models_{iF_k}$ semantics. To model-check an ATL formula φ we start by model-checking the innermost subformula ψ_1 with a strategy quantifier as main connective and label all states where it is satisfied with a new proposition symbol p_1 and replace ψ_1 with p_1 in φ. Note that ψ_1 is an ATL_1 formula. According to Corollary 13 this replacement does not change the truth value of the formula. We again take the innermost subformula ψ_2 with a strategy quantifier as main connective and perform the same procedure and keep doing the same thing until φ itself has finally been checked. Note that ψ_2 is also an ATL_1 formula after the replacement of ψ_1 with p_1. Since there can be at most a linear amount of calls to the NP-algorithm checking all the ATL_1 formulae, this problem is in $\Delta_2^p = P^{NP}$.

The Δ_2^p-hardness follows from the Δ_2^p-hardness of model-checking ATL with $\models_{ir}$ semantics and Proposition 23. For a proof of Δ_2^p-hardness in the memoryless case, see [JD08]. □

As in the case of ATL with incomplete information, we stay in the same complexity class when model-checking ATL^* with bounded-memory semantics as we do when model-checking with memoryless semantics.

Theorem 26 *Model-checking ATL^* with $\models_{iF_k}$ semantics is $PSPACE$-complete.*

Proof. Since model-checking ATL^* with $\models_{ir}$ semantics is $PSPACE$-complete, then by Proposition 23 we have that model-checking $ATL^*_{iF_k}$ is $PSPACE$-hard.

A $PSPACE$ algorithm for model-checking an $ATL^*_{iF_k}$ formula φ uses the same idea as the Δ_2^p algorithm for model-checking ATL with $\models_{iF_k}$ semantics. We

start with the innermost state formulae of φ and work our way out, at each point labelling states where the processed subformulae are true and treating them as proposition symbols from then on, just as when doing CTL^* model-checking. This replacement of state subformulae with proposition symbols is allowed because of Lemma 12. When the main connective of such a subformula is a strategy quantifier we non-deterministically guess a strategy σ as in the ATL_{iF_k} case and construct the same transition system TS_σ. However the formula to model-check in TS_σ is an LTL formula in this case. Each of the subformulae of φ can thus be checked in $NPSPACE = PSPACE$ and since there can only be a polynomial number of these, the model-checking of φ can be done in $PSPACE$. □

From this result we get the following

Corollary 27 *Model-checking ATL^* with $\models_{IF_k}$ semantics is $PSPACE$-complete.*

Proof. By Proposition 23 the problem is $PSPACE$-hard since it is at least as hard as model-checking ATL^* with $\models_{Ir}$ semantics which is $PSPACE$-hard.The problem is also in $PSPACE$ since it is included in model-checking of ATL^* with $\models_{iF_k}$ semantics which is in $PSPACE$. □

Thus, for both ATL/ATL^* and both complete/incomplete information we are in the same complexity classes when model-checking with bounded-memory semantics as with memoryless semantics. This is of course a very positive result, since bounded-memory makes us able to find winning strategies in many more games than if we just considered memoryless strategies. In addition, it is very positive that model-checking ATL^* with incomplete information and bounded-memory semantics is $PSPACE$-complete just as it is with complete information and memoryless semantics. For all the other cases we consider, model-checking with incomplete information is computationally harder than model-checking with complete information.

5.4 Finite-memory semantics

5.4.1 Complete information

For complete information games only finite-memory is needed for both ATL and ATL^* as was shown in the previous chapter. In fact, in ATL only memoryless strategies are needed. These observations leads to the following results.

Proposition 28 *Model-checking ATL with $\models_{IF}$ semantics is $PTIME$-complete*

Proof. From Corollary 15 follows that $M, s \models_{Ir} \varphi$ iff $M, s \models_{IF} \varphi$ for all CGMs $\mathcal{M}$, all states s and all ATL formulae φ. Since model-checking ATL with $\models_{Ir}$ semantics is P-complete [AHK02] so is model-checking ATL with $\models_{IF}$ semantics. □

Proposition 29 *Model-checking ATL^* with $\models_{IF}$ semantics is $2EXPTIME$-complete*

Proof. From Theorem 16 we have $M, \rho \models_{IF} \varphi$ iff $M, \rho \models_{IR} \varphi$ for all CGMs $\mathcal{M}$, all paths ρ and all ATL^* formulae φ. Since model-checking ATL^* with $\models_{IR}$ semantics is $2EXPTIME$-complete [AHK02] so is model-checking ATL^* with $\models_{IF}$ semantics. □

5.4.2 Incomplete information

We will now consider the case of finite-memory semantics and incomplete information. This problem was shown to be undecidable for perfect recall strategies. The result was mentioned in [AHK02] and referred to unpublished personal communication with M. Yannakakis. This was also referred to in [Sch04, JÅ07]. A proof was later published in [DT11], but the proof requires infinite memory strategies and the idea therefore does not seem to apply here. We will present a new proof which works both for perfect recall semantics and for finite-memory semantics. Here, the model-checking problem will be shown to be undecidable by a reduction from the halting problem of a Turing machine on an empty input-tape. The idea is roughly the same as used in [PR90, FS05, Sch08] but adjusted to the framework of concurrent game structures. Specifically, from a

given deterministic Turing machine T we construct an iCGM $\mathcal{M}_T = (\mathcal{G}_T, \pi_T)$ with 3 players, an ATL formula ψ_T and an initial state s_0 such that

$$T \text{ halts on the empty input if and only if } \mathcal{M}_T, s_0 \models_{iF} \psi_T$$

Thus, if the model-checking problem were decidable, then the halting problem would be decidable as well since we could solve the halting problem using an algorithm for the model-checking problem. Because we know that the halting problem is undecidable, such an algorithm cannot exist and we can therefore conclude that the model-checking problem is undecidable.

We begin with the construction of the iCGM $\mathcal{M}_T = (\mathcal{G}_T, \pi_T)$.

Construction of game model

The idea of the construction is illustrated in Figure 5.2 where two games G_1 and G_2 are played at the same time synchronously, starting from the state (idle, idle). The states of the game structure $\mathcal{G}_T$ will be pairs, where the first element is a state from G_1 and the second element is a state from G_2.

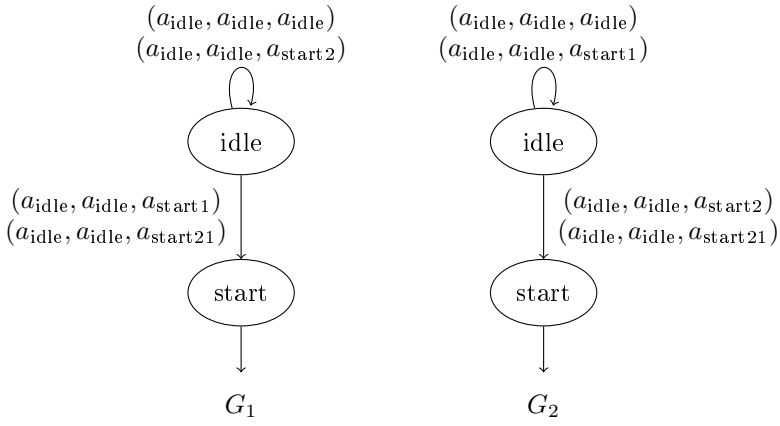

Figure 5.2: Illustration of concurrent game structure $\mathcal{G}$

The game is played as follows: In both G_1 and G_2, player 3 controls when (if ever) a start state is reached, which is the only thing that player 3 controls. He has complete information of everything that happens. When a start state has

been reached in G_1, player 1 controls the play in that module from that point on, whereas player 2 controls the play in G_2 when the start state in that module has been reached. However, player 1 does not know what happens in the G_2 and player 2 does not know what happens in G_1 (in fact, player $j \in \{1,2\}$ can only observe if the play is in the idle state of G_j or not). Each of the modules G_j consists of one state for each control state of T, one state for each tape symbol of T and one special state \$. At every point in time (after the start state) player $j = 1, 2$ can choose any of these states as the successor state in module G_j. The common objective of player 1 and 2 will be designed so they have a sure-winning strategy if and only if they both play according to the run of T starting with the blank tape and such that they both eventually reach the halting state, given that player 3 eventually chooses the start state in each module (though not necessarily at the same time). What we mean by playing according to the run of T will be specified in more detail later. However, informally player 1 and 2 will play one configuration of T at a time, given some specific encoding, and the configurations will be separated by choosing the \$ state. We will show that player 1 and 2 can make sure that a common objective captured by the LTL formula φ_T is satisfied if and only if the Turing machine halts on the empty input. And by implication that model-checking $\psi_T = \langle\langle\{1,2\}\rangle\rangle\varphi_T$ in the initial state (idle, idle) with $\models_{iF}$ semantics corresponds to deciding whether T halts on the empty input tape. Roughly, this is because if player 1 and 2 have a sure-winning strategy, then they have a finite-memory sure-winning strategy because of the finite run of T in the case that T halts on an empty input tape.

More formally, let $T = (Q, \Sigma, \delta, q_0, B, \text{halt})$ be a deterministic Turing machine where Q is a finite set of control states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, R\}$ is a transition function, $q_0 \in Q$ is the initial control state, $B \in \Sigma$ is the blank symbol and halt $\in Q$ is the halting state. We assume without loss of generality that T never writes the blank symbol and has a semi-infinite tape, i.e. it never moves to the left of the initial tape position. Let $\mathcal{G}_T = (\text{States}, \{1,2,3\}, \text{Act}, \text{Mov}, \text{Tab}, (\sim_1, \sim_2, \sim_3))$ be constructed from T such that

- $\text{States} = (\{\text{idle}, \text{start}, \$\} \cup \Sigma \cup Q)^2$
- $\text{Act} = \{a_{\text{idle}}, a_{\text{start}1}, a_{\text{start}2}, a_{\text{start}12}, a_\$\} \cup \{a_q | q \in Q\} \cup \{a_s | s \in \Sigma\}$

The equivalence relations $\sim_j$ for player $j = 1, 2$ are defined so they only observe if the game is in the start state in G_j or not and no information about G_{3-j}.

$$(s_1, s_2) \sim_j (s_1', s_2') \text{ if and only if } s_j, s_j' \neq \text{idle} \vee s_j = s_j' = \text{idle}$$

whereas player 3 has complete information:

$$(s_1, s_2) \sim_3 (s_1', s_2') \text{ if and only if } s_1 = s_1' \wedge s_2 = s_2'$$

Now for player $j = 1, 2$ define the legal moves as

$$\text{Mov}((x_1, x_2), j) = \begin{cases} \{a_{\text{idle}}\} & \text{if } x_j = \text{idle} \\ \{a_q | q \in Q\} \cup \{a_s | s \in \Sigma\} \cup \{a_\$\} & \text{if } x_j \neq \text{idle} \end{cases}$$

and for player 3 define the legal moves as

$$\text{Mov}((x_1, x_2), 3) = \begin{cases} \{a_{\text{idle}}, a_{\text{start}1}, a_{\text{start}2}, a_{\text{start}12}\} & \text{if } x_1, x_2 = \text{idle} \\ \{a_{\text{idle}}, a_{\text{start}1}\} & \text{if } x_1 = \text{idle and } x_2 \neq \text{idle} \\ \{a_{\text{idle}}, a_{\text{start}2}\} & \text{if } x_2 = \text{idle and } x_1 \neq \text{idle} \\ \{a_{\text{idle}}\} & \text{if } x_1, x_2 \neq \text{idle} \end{cases}$$

Finally, define Tab for all action tuples (a_x, a_y, a_z) legal in state (x_1, x_2) as

$$\text{Tab}((x_1, x_2), (a_x, a_y, a_z)) = \begin{cases} (x, y) & \text{if } x_1, x_2 \neq \text{idle} \\ (z, y) & \text{if } x_1 = \text{idle}, \ x_2 \neq \text{idle} \\ (x, z) & \text{if } x_2 = \text{idle}, \ x_1 \neq \text{idle} \\ (z, z) & \text{if } x_1, x_2 = \text{idle} \end{cases}$$

We let the finite set Prop of proposition symbols be equal to States and define the labeling function π such that $\pi(s) = \{s\}$ for all $s \in \text{States}$.

Construction of ATL^* formula

We have now defined the game structure $\mathcal{M}$ and next we will define the ATL formula $\langle\langle\{1, 2\}\rangle\rangle\varphi_T$ with the properties outlined in the beginning. Since the LTL formula φ is quite large we start by defining subformulae of it with the following meaning, where $j = 1, 2$

- $\varphi_{start}(j)$: The play eventually reaches start in G_j

- $\varphi_{halt}(j)$: The play will eventually reach the halting state in G_j
- $\varphi_{init}(j)$: Player j plays according to the first two configurations of T immediately after start is reached in G_j.
- $\varphi_{succ}(j)$: If the play is in $(\$, \$)$ and player j and $3-j$ afterwards play configurations C_j and C_{3-j} of T respectively such that $C_j \vdash C_{3-j}$, then they must continue by playing two configurations C'_j and C'_{3-j} such that $C'_j \vdash C'_{3-j}$. Here $C \vdash D$ for two configurations C and D means that D is a successor configuration of C for T. For any halting configuration C_H we use the convention that $C_H \vdash C_H$.

By playing a configuration C of a Turing machine with control state q, tape head at position r and with the non-blank symbols on the tape being $a_1, ..., a_k$ we mean playing the following sequence of states in G_j

$$a_1 ... a_{r-1} q a_r ... a_k$$

That is, all non-blank tape symbols are played as a left-to-right scan of the tape of the Turing machine, but with the control state played just before the content of the tape cell that is pointed to by the tape head. We call this sequence ρ_C for configuration C and the length of it $|\rho_C|$.

We first present $\varphi_{start}(1), \varphi_{init}(1)$ and $\varphi_{halt}(1)$. Let $\text{States}_1 = \{\text{idle}, \text{start}, \$\} \cup \Sigma \cup Q$, i.e. the set of all states in G_1 or G_2. Then we have

$$\varphi_{start}(1) = \bigvee_{x \in \text{States}_1} \mathbf{F}(\text{start}, x)$$

$$\varphi_{halt}(1) = \bigvee_{x \in \text{States}_1} \mathbf{F}(\text{halt}, x)$$

$$\varphi_{init}(1) = (\bigvee_{x \in \text{States}_1} (\text{start}, x)) \rightarrow$$

$$\mathbf{X} \left(\bigvee_{x \in \text{States}_1} (\$, x) \right) \wedge$$

$$\mathbf{X}^2 \left(\bigvee_{x \in \text{States}_1} (q_0, x) \right) \wedge$$

$$\mathbf{X}^3 \left(\bigvee_{x \in \text{States}_1} (\$, x) \right) \wedge$$

$$\mathbf{X}^4 \left(\bigvee_{x \in \text{States}_1} (\delta_2(q_0, B, R), x) \right) \wedge$$

$$\mathbf{X}^5 \left(\bigvee_{x \in \text{States}_1} (\delta_1(q_0, B, R), x) \right) \wedge$$

$$\mathbf{X}^6\left(\bigvee_{x\in \mathrm{States}_1}(\$,x)\right)$$

Note that since the tape is semi-infinite the first move of the tape head must be a right move, which keeps $\varphi_{init}(1)$ simpler. This means that to satisfy $\varphi_{init}(1)$, player 1 must choose the following sequence of actions after the start state is reached in G_1

$$a_{\$},\quad a_{q_0},\ a_{\$},\ a_{\delta_2(q_0,B,R)},\ a_{\delta_1(q_0,B,R)},\ a_{\$}$$

We obtain $\varphi_{start}(2), \varphi_{halt}(2)$ and $\varphi_{init}(2)$ from $\varphi_{start}(1), \varphi_{init}(1)$ and $\varphi_{halt}(1)$ by replacing every proposition (x,y) with (y,x).

We look at a number of cases when designing $\varphi_{succ}(j)$ for $j=1,2$. As for the other formulae we first focus on $\varphi_{succ}(1)$ and then obtain $\varphi_{succ}(2)$ from it by using the same substitution. First note that if two configurations C and C' of T are such that $C \vdash C'$ then $|\rho_C| = |\rho_{C'}|$ unless the move from C to C' is a move where the tape head points to a cell with the blank symbol before the move (which must be the cell to the right of the rightmost cell containing a non-blank symbol according to the rules of the Turing machine). In this case $|\rho_C| + 1 = |\rho_{C'}|$.

We first introduce the LTL formula φ_L which is true on a path

$$\alpha = (s_1, s_1')(s_2, s_2')...(s_k, s_k')(\$,\$)\alpha'$$

for any suffix α' if $(s_1, ..., s_k)$ encodes a configuration C of T and $(s_1', ..., s_k')$ encodes a configuration C' such that $C \vdash C'$ and that it is a left-move that leads from configuration C to C'. This is illustrated in Figure 5.3 where members of States are written as column vectors for clarity, so symbols on top are played by player 1 and symbols at the bottom are played by player 2. Note that this does not cover the situation in which $|\rho_C| \neq |\rho_{C'}|$, which we handle separately in the formula φ_{LB}. φ_L is now defined by

$$\varphi_L = \left[\bigvee_{s\in\Sigma}(s,s)\right]\mathbf{U}$$

$$\left[\bigvee_{q\in Q}\bigvee_{s\in\Sigma}\bigvee_{s'\in\Sigma}\left((s',\delta_1(q,s))\wedge\mathbf{X}(q,s')\wedge\mathbf{XX}(s,\delta_2(q,s))\right)\wedge\right.$$

$$\left.\mathbf{XXX}\left(\left(\bigvee_{s\in\Sigma}(s,s)\right)\mathbf{U}(\$,\$)\right)\right]$$

In the same way φ_R encodes that the two players play two configurations C and C' where $C \vdash C'$ by a right-move of the Turing machine where $|\rho_C| = |\rho_{C'}|$.

$$\varphi_R = \left[\bigvee_{s\in\Sigma}(s,s)\right]\mathbf{U}$$

$$[\bigvee_{q\in Q}\bigvee_{s\in\Sigma}((q,\delta_2(q,s))\wedge\mathbf{X}(s,\delta_1(q,s)))\wedge$$

$$\mathbf{XX}\left((\bigvee_{s\in\Sigma}(s,s))\,\mathbf{U}(\$,\$)\right)]$$

We also have φ_H which encodes that the two players both play the same halting configuration C.

$$\varphi_H = \left[\bigvee_{s\in\Sigma}(s,s)\right]\mathbf{U}\left[(\text{halt},\,\text{halt})\wedge\mathbf{X}\left((\bigvee_{s\in\Sigma}(s,s))\,\mathbf{U}(\$,\$)\right)\right]$$

$$\alpha_L = \binom{s_1}{s_1}\binom{s_2}{s_2}\ldots\binom{s_{r-1}}{\delta_1(q,s_r)}\binom{q}{s_{r-1}}\binom{s_r}{\delta_2(q,s_r)}\binom{s_{r+1}}{s_{r+1}}\ldots\binom{s_m}{s_m}\binom{\$}{\$}$$

$$\alpha_R = \binom{s_1}{s_1}\binom{s_2}{s_2}\ldots\binom{q}{\delta_2(q,s_r)}\binom{s_r}{\delta_1(q,s_r)}\binom{s_{r+1}}{s_{r+1}}\ldots\binom{s_m}{s_m}\binom{\$}{\$}$$

$$\alpha_H = \binom{s_1}{s_1}\binom{s_2}{s_2}\ldots\binom{s_{r-1}}{s_{r-1}}\binom{\text{halt}}{\text{halt}}\binom{s_{r+1}}{s_{r+1}}\ldots\binom{s_m}{s_m}\binom{\$}{\$}$$

Figure 5.3: Illustration of subplays each corresponding to a move of the Turing machine. A left move is on top, a right move in the middle and a "halting move" at the bottom. It is assumed that all $s_i \in \Sigma$ and $q, q' \in Q$.

In addition to these basic three moves, we need to be able to encode the case of moves where the tape head is pointing to a cell containing a blank symbol before the move, since this is the only case in which $C \vdash C'$ where $|\rho_C| \neq |\rho_{C'}|$. Since the Turing machine never writes the blank symbol and has a semi-infinite tape, this can only occur when the tape head is just to the right of the rightmost non-blank symbol. This situation is specified by the formulae φ_{LB} and φ_{RB}:

$$\varphi_{LB} = \left[\bigvee_{s\in\Sigma}(s,s)\right]\mathbf{U}$$

$$[\bigvee_{q\in Q}\bigvee_{s'\in\Sigma}((s',\delta_1(q,B))\wedge\mathbf{X}(q,s')\wedge\mathbf{XX}(\$,\delta_2(q,B)))\wedge$$

$$\mathbf{XXX}\left(\bigvee_{t\in\Sigma\cup Q}(t,\$)\right)]$$

$$\varphi_{RB} = \left[\bigvee_{s\in\Sigma}(s,s)\right]\mathbf{U}$$

$$\left[\bigvee_{q\in Q}((q,\delta_2(q,B))\wedge\mathbf{X}(\$,\delta_1(q,B)))\wedge\mathbf{XX}\left(\bigvee_{t\in\Sigma\cup Q}(t,\$)\right)\right]$$

Since we need $\varphi_{succ}(1)$ to express that if the two players play two configurations C_1 and C_2 starting at the same time such that $C_1 \vdash C_2$ then they must continue (after \$) by playing configurations C_1' and C_2' such that $C_1' \vdash C_2'$. This is simple enough to express given our subformulae above in most cases, but gives a little difficulty in the case where $|\rho_{C_1}| < |\rho_{C_2}|$. This is because when playing the configurations C_1' and C_2', player 1 starts one time step before player 2. This means that we need formulae for left moves, right moves, halting moves, left moves overwriting a blank symbol and right moves overwriting a blank symbol in the case where the players start with an offset of one step. These formulae will be written with a superscript $+$, for instance a normal left move with an offset of one step will be expressed by the formula φ_L^+. The construction of $\varphi_L^+, \varphi_R^+, \varphi_H^+, \varphi_{LB}^+$ and φ_{RB}^+ can be found in Appendix B.

We now have the subformulas needed to express $\varphi_{succ}(1)$. This is done as follows:

$$\varphi_{succ}(1) = \mathbf{G}\Big[(\$,\$) \rightarrow$$

$$\Big((\mathbf{X}(\varphi_L \vee \varphi_R \vee \varphi_H) \rightarrow \mathbf{X}(\neg(\$,\$)\ \mathbf{U}\ ((\$,\$) \wedge \mathbf{X}(\varphi_L \vee \varphi_R \vee \varphi_H \vee \varphi_{LB} \vee \varphi_{RB}))))\wedge$$

$$(\mathbf{X}(\varphi_{LB} \vee \varphi_{RB}) \rightarrow \mathbf{X}(\textstyle\bigwedge_{t \in \Sigma \cup Q} \neg(\$,t)\ \mathbf{U}\ (\bigvee_{t \in \Sigma \cup Q}(\$,t) \wedge \mathbf{X}(\varphi_L^+ \vee \varphi_R^+ \vee \varphi_H^+ \vee \varphi_{LB}^+ \vee \varphi_{RB}^+)))))\Big)\Big]$$

This formula specifies that it is always the case that if the two players play two configurations C_1 and C_2 of T such that $C_1 \vdash C_2$ starting at the same time, then they must continue by playing two configurations C_1' and C_2' such that $C_1' \vdash C_2'$ afterwards.

The formula $\varphi_{succ}(2)$ is obtained from $\varphi_{succ}(1)$ by replacing every proposition (x, y) with (y, x).

The ATL^* formula we wish to model-check is $\psi_T = \langle\!\langle \{1,2\} \rangle\!\rangle \varphi_T$ where φ_T is defined as

$$\varphi_T = \Big(\textstyle\bigwedge_{j \in \{1,2\}} \varphi_{start}(j)\Big) \rightarrow \bigwedge_{j \in \{1,2\}} \varphi_{halt}(j) \wedge \varphi_{init}(j) \wedge \varphi_{succ}(j)$$

We will show later that if player 1 and 2 have a collective sure-winning strategy for this objective in $\mathcal{M}_T$, then they must both play exactly according to the rules of the Turing machine whenever the play starts in G_i. Otherwise, player 3 will have a spoiler strategy that makes either $\varphi_{init}(j)$ or $\varphi_{succ}(j)$ false for some $j \in \{1,2\}$. In addition, in a sure-winning strategy both players must also reach a halting state, given that a start state is reached. Put together, this means that if the Turing machine T does not halt on the empty input tape, then player 1 and

2 does not have a sure-winning strategy for φ_T, because they cannot both follow the rules of a non-halting Turing machine and reach a halting state. On the other hand, we will show that if the Turing machine actually halts, then player 1 and 2 have a collective sure-winning strategy which consists in them both following the moves of the Turing machine when the play starts in their own component of the game. We will also show that such a strategy can be implemented with finite memory. These are the main arguments that $\mathcal{M}_T, (\text{idle}, \text{idle}) \models_{iF} \psi_T$ if and only if T halts on the empty input tape.

THEOREM 30 *Model-checking ATL^*_{iF} is undecidable for $n \geq 3$ players.*

PROOF. We wish to show that $\mathcal{M}_T, (\text{idle,idle}) \models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$ if and only if T halts on the empty input which implies that model-checking ATL^*_{iF} is undecidable, since the halting problem for a deterministic Turing machine on an empty input tape with a semi-infinite tape that never writes the blank symbol is undecidable [HMU03].

First, assume that T halts on the empty input. In this case player 1 and 2 can play according to strategies σ_1 and σ_2 where they play the configurations of the unique run of T if player 3 moves the play to start in G_1 and G_2 respectively. This strategy only needs finite memory, since T halts in a finite number of steps and after having played this finite number of finite configurations, the players only need to remember the finite configuration of T at the point of halting. Thus, they must both have enough memory to remember the run of the Turing machine. Suppose for contradiction that player 3 has a spoiling strategy σ_3 against player 1 and 2 using strategies σ_1 and σ_2 respectively. First, the play must reach start in both G_1 and G_2, otherwise φ_T is satisfied. Then, since both player 1 and 2 will eventually play a_{halt}, $\varphi_{halt}(1)$ and $\varphi_{halt}(2)$ are satisfied. In addition $\varphi_{init}(1)$ and $\varphi_{init}(2)$ are also satisfied since player 1 and 2 play according to the run of T. $\varphi_{succ}(1)$ is also satisfied for any choice of σ_3, because whenever the play reaches $(\$, \$)$ and player 1 and 2 afterwards play configurations C_1 and C_2 such that $C_1 \vdash C_2$, then player 1 will continue by playing C_2 which is the next configuration of T and player 2 will continue by playing the next configuration C_3 of T, which is such that $C_2 \vdash C_3$. By symmetry, $\varphi_{succ}(2)$ is also satisfied for any choice of σ_3. Thus, σ_3 cannot be a spoiling strategy for σ_1 and σ_2 and therefore $\mathcal{M}_T, (\text{idle,idle}) \models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$ if T halts on the empty input.

Next, assume that T does not halt on the empty input. We will show that for all strategies σ_1 and σ_2 for player 1 and 2, player 3 has a spoiling strategy σ_3. Assume for contradiction that there exists strategies σ_1 and σ_2 for player 1 and 2 such no such spoiling strategy exists. Let us denote by $C_1, C_2, ...$ the configurations of the run of T. First, we note that immediately after start is

played in G_j then player j must play the first two configurations C_1 and C_2 of T, otherwise player 3 has a spoiling strategy to falsify $\varphi_{init}(j)$ for some $j \in \{1,2\}$.

We will proceed by an inductive argument to show that the two players must keep playing the correct configurations of the Turing machine using this as a base case. Player 3 can start the play in both components such that player 1 plays C_1 starting at the same time as player 2 starts playing C_2. Then, since player 1 must continue by playing C_2 (to satisfy $\varphi_{init}(1)$), player 2 must continue by playing C_3 after he has played C_2 (because otherwise $\varphi_{succ}(1)$ is falsified). And he has to do this no matter when the play is started by player 3 in G_2, since he cannot know when (and if) the play is started in G_1. If he doesn't then player 3 has a counter strategy. The same argument can be used to show that player 1 must play C_3 after C_2 by symmetry. We can then proceed by the same argument to show that player 2 must play C_4 after playing C_3 no matter when the game is started, because otherwise player 3 has a spoiling strategy that consists in starting the play such that player 1 starts playing C_2 at the same time as player 2 starts playing C_3. And since player 1 continues by playing C_3, player 2 must continue by playing C_4 to make sure $\varphi_{succ}(1)$ is not falsified. By symmetry, player 1 must also play C_4 after C_3 no matter when the play has started, and so on. This argument can be extended to show that player 1 and 2 must keep playing according to the rules of T indefinitely, because otherwise player 3 has a spoiling strategy that falsifies $\varphi_{init}(j)$ or $\varphi_{succ}(j)$ for some $j \in \{1,2\}$.

Thus, player 1 and 2 must play according to the rules of T in the sure-winning strategies σ_1 and σ_2. However, a winning play must satisfy $\varphi_{halt}(j)$ for $j = 1, 2$ which is not the case for a play of σ_1 and σ_2 since they play according to the Turing machine T that does not halt. This means that σ_1 and σ_2 are not sure-winning and we have a contradiction. Thus, for all collective strategies σ_1 and σ_2 for player 1 and 2, there exists a spoiling strategy for player 3. Moreover, this holds for all finite-memory strategies σ_1 and σ_2 as well since finite-memory strategies are included in the set of strategies. Therefore $\mathcal{M}_T, (\text{idle,idle}) \not\models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$ if T does not halt on the empty input.

In total, we have shown that $\mathcal{M}_T, (\text{idle,idle}) \models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$ if and only if T halts on the empty input. This means that model-checking ATL^* with incomplete information and finite-memory semantics is undecidable since the halting problem for Turing machines is undecidable. The same proof can be used to show that model-checking ATL^* with traditional $\models_{iR}$ semantics is undecidable. □

The construction used in the proof of undecidability can be altered to show the following result

Theorem 31 $\mathbb{M}_{iF}^{ATL^*} \subset \mathbb{M}_{iR}^{ATL^*}$

Proof. Consider the the same iCGM $\mathcal{M}_T$ from the proof of Theorem 30, but now consider the following formula φ_T where player 1 and 2 are still required to play according to the rules of the Turing machine T in order to have a sure-winning strategy, but where they also need the Turing machine not to halt in order to win.

$$\varphi_T = \left(\bigwedge_{j \in \{1,2\}} \varphi_{start}(j) \right) \rightarrow \bigwedge_{j \in \{1,2\}} \neg\varphi_{halt}(j) \wedge \varphi_{init}(j) \wedge \varphi_{succ}(j)$$

We will show that there exists a Turing machine T such that

$$\mathcal{M}_T, (\text{idle, idle}) \models_{iR} \langle\langle\{1,2\}\rangle\rangle\varphi_T \text{ but } \mathcal{M}_T, (\text{halt, halt}) \not\models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$$

Consider the Turing machine T that prints the symbol d and moves right at every step without ever halting. In order for player 1 and 2 to win, they must play the following sequence of symbols when the play starts in their component in order to be sure-winning:

$$\$q_0\$dq_0\$ddq_0\$dddq_0\$ddddq_0...$$

Suppose for contradiction that there is such a k-memory strategy for some k for the two players. At every step after leaving the idle state, the players will be in the same information set and thus the input to the Mealy machine corresponding to the k-memory strategies will be the same at every time step after they start printing the configurations of T. At some point each player will have to play $k+1$ sequences of d's followed by q_0, however as in the proof of Proposition 6 this is not possible for a k-memory strategy which receives the same input symbol at each step. Thus, there exists no sure-winning k-memory strategy for the two players and thus $\mathcal{M}_T, (\text{idle, idle}) \not\models_{iF} \langle\langle\{1,2\}\rangle\rangle\varphi_T$.

However, there exists an infinite memory strategy σ_j for player $j \in \{1,2\}$ playing this sequence defined by

$$\sigma_j(I_{j,idle}^{c_1} I_{j,nidle}^{c_2}) = \begin{cases} a_{idle} & \text{if } c_2 = 0 \\ a_{q_0} & \text{if } c_2 = \frac{(n+1)\cdot(n+2)}{2} - 1 \text{ for some } n \in \mathbb{N} \\ a_\$ & \text{if } c_2 = \frac{(n+1)\cdot(n+2)}{2} \text{ for some } n \in \mathbb{N} \cup \{0\} \\ a_d & \text{if } \frac{(n+1)\cdot(n+2)}{2} < c_2 < \frac{(n+2)\cdot(n+3)}{2} - 1 \text{ for some } n \in \mathbb{N} \end{cases}$$

where $I_{j,idle}$ is the information set for player j containing the idle state and $I_{j,nidle}$ is the other information set. This means that if the players can always remember how long time went by since they left the idle state, they can use this to obtain a winning strategy. However, this requires infinite memory. Thus $\mathcal{M}, (\text{idle}, \text{idle}) \models_{iR} \langle\langle\{1,2\}\rangle\rangle\psi$. □

It is quite interesting to see that infinite memory is needed in some cases with incomplete information in order to have sure-winning strategies. We still don't know if this is the case for ATL however. It was the hope that we would find decidable model-checking problems by restricting to finite-memory strategies, but unfortunately this was not the case for ATL^*. However, one gets decidability if the memory size is bounded. This is a good result, because in many applications of model-checking for program synthesis, one can give a good estimate of how much memory is available for a given implementation and then use the available memory size as input for the bounded-memory model-checking algorithm.

Chapter 6

Conclusion

We have introduced concurrent game structures and motivated the use of the notion for modelling the behavior of reactive systems. Next, we introduced alternating-time temporal logic and motivated its use in program synthesis when interpreted over concurrent game models. We proceeded by motivating the analysis of finite-memory strategies when model-checking ATL/ATL^* formulae with the hope of obtaining lower complexity than with perfect recall strategies, but without compromising too much on the expressiveness. Our main results can be found in Figure 4.1 and 5.1. We have presented both positive and negative results in this thesis. The positive results are that bounded-memory semantics does not increase the complexity too much compared to memoryless semantics, but makes it possible to solve many more games. It is also a positive result that the complexity of ATL^* is still $PSPACE$-complete with bounded-memory semantics and incomplete information as it is with memoryless semantics and complete information. On the other hand, it is a negative result that with unbounded memory the complexity of the model-checking problem is approximately as high as for perfect recall semantics in all the cases that we solved. The problems that we solved were done in many cases by reusing techniques from model-checking the logics LTL, CTL and CTL^* as well as techniques for model-checking ATL/ATL^* with memoryless and perfect recall semantics. In addition, results and algorithms for sure-winning strategies in two-player turn-based games proved to be very useful in the complete information case. There are still a number of open questions, however. It still remains an open

question whether model-checking ATL with finite-memory semantics and incomplete information is decidable, but we have found exact complexity results for all the other cases considered. This question of decidability is also related to the question of whether finite-memory semantics is the same as perfect recall semantics for ATL with incomplete information. This was shown not to be the case for ATL^* where infinite memory is needed in some games. It would also be interesting to investigate whether ATL/ATL^* model-checking is decidable for two-player games, since both our proof and the proof in [DT11] are for three or more players.

Appendix A

Proof of Lemma 9

In order to prove Lemma 9 we first prove the following

Lemma 32 *For every LTL formula φ over some finite alphabet Σ generated by the grammar*

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \boldsymbol{X}\boldsymbol{X}\varphi_1 \mid \varphi_1 \boldsymbol{U} \varphi_2$$

where $p \in \Sigma$ is a proposition symbol and φ_1 and φ_2 are generated by the same grammar we have

$$a_1 a_1 a_2 a_2 ... \models \varphi \text{ iff } a_1 a_2 a_2 a_3 a_3 ... \models \varphi$$

for all sequences of the form $a_1 a_1 a_2 a_2 ... \in (2^{\Sigma})^{\omega}$

Proof. The proof is by induction on the structure of φ. For the base case we have

$\varphi = p$:

For all sequences of the form $a_1a_1a_2a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_1a_2a_2... \models \varphi \text{ iff } p \in a_1 \text{ iff } a_1a_2a_2a_3a_3... \models \varphi$$

As induction hypothesis suppose the lemma holds for all subformulae of φ. The different cases are as follows

$\varphi = \neg\varphi_1$:

For all sequences of the form $a_1a_1a_2a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_1a_2a_2... \models \varphi \text{ iff } a_1a_1a_2a_2... \not\models \varphi_1 \text{ iff}$$

$$a_1a_2a_2a_3a_3... \not\models \varphi_1 \text{ iff } a_1a_2a_2a_3a_3... \models \varphi$$

by using the induction hypothesis.

$\varphi = \varphi_1 \vee \varphi_2$:

For all sequences of the form $a_1a_1a_2a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_1a_2a_2... \models \varphi \text{ iff } a_1a_1a_2a_2... \models \varphi_1 \text{ or } a_1a_1a_2a_2... \models \varphi_2 \text{ iff}$$

$$a_1a_2a_2a_3a_3... \models \varphi_1 \text{ or } a_2a_2a_3a_3... \models \varphi_2 \text{ iff } a_1a_2a_2a_3a_3... \models \varphi$$

by using the induction hypothesis.

$\varphi = \mathbf{XX}\varphi_1$:

For all sequences of the form $a_1a_1a_2a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_1a_2a_2... \models \varphi \text{ iff } a_2a_2a_3a_3... \models \varphi_1 \text{ iff}$$

$$a_2a_3a_3a_4a_4... \models \varphi_1 \text{ iff } a_1a_2a_2a_3a_3... \models \varphi$$

by using the induction hypothesis.

$\varphi = \varphi_1\mathbf{U}\varphi_2$:

For all sequences of the form $a_1a_1a_2a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_1a_2a_2... \models \varphi_1\mathbf{U}\varphi_2 \text{ iff}$$

$$a_1a_1a_2a_2... \models \varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1\mathbf{U}\varphi_2)) \text{ iff}$$

$a_1a_1a_2a_2... \models \varphi_2$ or $(a_1a_1a_2a_2... \models \varphi_1 \wedge \mathbf{X}(\varphi_1\mathbf{U}\varphi_2))$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2$ or $(a_1a_1a_2a_2... \models \varphi_1$ and $a_1a_1a_2a_2... \models \mathbf{X}(\varphi_1\mathbf{U}\varphi_2))$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2$ or $(a_1a_2a_2a_3a_3... \models \varphi_1$ and $a_1a_2a_2a_3a_3... \models \varphi_1\mathbf{U}\varphi_2)$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2$ or $a_1a_2a_2a_3a_3... \models \varphi_1 \wedge \varphi_1\mathbf{U}\varphi_2$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2 \vee (\varphi_1 \wedge \varphi_1\mathbf{U}\varphi_2)$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2 \vee (\varphi_1 \wedge \varphi_1\mathbf{U}\varphi_2 \wedge \neg\varphi_2)$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1\mathbf{U}\varphi_2) \wedge \neg\varphi_2)$ iff

$a_1a_2a_2a_3a_3... \models \varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1\mathbf{U}\varphi_2))$ iff

$a_1a_2a_2a_3a_3... \models \varphi_1\mathbf{U}\varphi_2$

by using the induction hypothesis. □

We are now ready to prove

Lemma 9 *For all LTL formulae φ and $\varphi' = \varphi[\boldsymbol{X} \mapsto \boldsymbol{XX}]$ over some finite alphabet Σ it holds that for all sequences $a_1a_2a_3... \in (2^\Sigma)^\omega$*

$$a_1a_2a_3... \models \varphi \text{ iff } a_1a_1a_2a_2a_3a_3... \models \varphi'$$

Proof. Let φ be be interpreted over words in $(2^\Sigma)^\omega$ for some finite alphabet Σ. For any LTL formula ψ we define $\psi' = \psi[\mathbf{X} \mapsto \mathbf{XX}]$. We do the proof by induction on the structure of φ. As the base case φ is a proposition symbol

$\varphi = p$:

For all $a_1a_2... \in (2^\Sigma)^\omega$ we have

$$a_1a_2a_3... \models \varphi \text{ iff } p \in a_1 \text{ iff } a_1a_1a_2a_2... \models \varphi'$$

As induction hypothesis suppose that the Lemma holds for all subformulae φ_1 and φ_2 of φ. We have the following cases

$\varphi = \neg\varphi_1$:

For all $a_1a_2... \in (2^\Sigma)^\omega$ we have

$a_1a_2a_3... \models \varphi$

iff $a_1a_2a_3... \not\models \varphi_1$

iff $a_1a_1a_2a_2... \not\models \varphi_1'$ (by the induction hypothesis)

iff $a_1a_1a_2a_2... \models \varphi'$

since $\varphi' = \neg\varphi_1'$

$\varphi = \varphi_1 \vee \varphi_2$:

For all $a_1a_2... \in (2^\Sigma)^\omega$ we have

$a_1a_2a_3... \models \varphi$

iff $a_1a_2a_3... \models \varphi_1$ or $a_1a_2a_3... \models \varphi_2$

iff $a_1a_1a_2a_2... \models \varphi_1'$ or $a_1a_1a_2a_2... \models \varphi_2'$ (by the induction hypothesis)

iff $a_1a_1a_2a_2... \models \varphi'$

since $\varphi' = \varphi_1' \vee \varphi_2'$

$\varphi = \mathbf{X}\varphi_1$:

For all $a_1a_2... \in (2^\Sigma)^\omega$ we have

$a_1a_2a_3... \models \varphi$

iff $a_2a_3... \models \varphi_1$

iff $a_2a_2a_3a_3... \models \varphi_1'$ (by the induction hypothesis)

iff $a_1a_1a_2a_2... \models \varphi'$

since $\varphi' = \mathbf{XX}\varphi_1'$

$\varphi = \varphi_1\mathbf{U}\varphi_2$:

For all $a_1a_2... \in (2^\Sigma)^\omega$ we have

$a_1a_2a_3... \models \varphi$

iff $\exists k.a_ka_{k+1}... \models \varphi_2$ and $\forall j < k.a_ja_{j+1}... \models \varphi_1$

iff $\exists k.a_ka_ka_{k+1}a_{k+1}... \models \varphi_2'$ and $\forall j < k.a_ja_ja_{j+1}a_{j+1}... \models \varphi_1'$

By using the fact that Lemma 32 applies to φ_1' the above is equivalent to

iff $\exists k.a_ka_ka_{k+1}a_{k+1}... \models \varphi_2'$ and

$\forall j < k.a_ja_ja_{j+1}a_{j+1}... \models \varphi_1' \wedge a_ja_{j+1}a_{j+1}... \models \varphi_1'$

iff $a_1a_1a_2a_2... \models \varphi_1' \mathbf{U} \varphi_2'$

which finishes the proof. □

Appendix B

Theorem 30 proof details

Below we define the formulae $\varphi_L^+, \varphi_R^+, \varphi_H^+, \varphi_{LB}^+$ and φ_{RB}^+

$$\varphi_L^+ = \Big\{ \bigvee_{s_1,s_2,s_3 \in \Sigma} \bigvee_{q \in Q} \Big((s_1, \$) \wedge \mathbf{X}(q, \delta_1(q, s_2)) \wedge \mathbf{X}^2(s_2, s_1) \wedge$$

$$\mathbf{X}^3 \Big\{ \Big((\$, \delta_2(q, s_2)) \wedge \mathbf{X} \bigvee_{t \in \Sigma \cup Q} (t, \$) \Big) \vee \Big((s_3, \delta_2(q, s_2)) \wedge$$

$$\Big[\bigvee_{s_5,s_6,s_7 \in \Sigma} (s_5, s_6) \wedge \mathbf{X}(s_7, s_5) \Big] \mathbf{U}$$

$$\Big[\bigvee_{s_5,s_6 \in \Sigma} \bigvee_{t \in \Sigma \cup Q} (s_5, s_6) \wedge \mathbf{X}(\$, s_5) \wedge \mathbf{X}^2(t, \$) \Big] \Big) \Big\} \Big\} \vee$$

$$\Big\{ \bigvee_{s,s' \in \Sigma} ((s, \$) \wedge \mathbf{X}(s', s)) \wedge$$

$$\mathbf{X} \Big(\Big[\bigvee_{s_1,s_2,s_3 \in \Sigma} ((s_1, s_2) \wedge \mathbf{X}(s_3, s_1)) \Big] \mathbf{U}$$

$$\Big[\bigvee_{s_1,\ldots,s_4 \in \Sigma} \bigvee_{q \in Q} \Big((s_1, s_2) \wedge \mathbf{X}(q, \delta_1(q, s_3)) \wedge \mathbf{X}^2(s_3, s_1) \wedge$$

$$\mathbf{X}^3 \Big\{ \Big((\$, \delta_2(q, s_3)) \wedge \mathbf{X} \bigvee_{t \in \Sigma \cup Q} (t, \$) \Big) \vee \Big((s_4, \delta_2(q, s_3)) \wedge$$

$$\left[\bigvee_{s_5,s_6,s_7\in\Sigma}(s_5,s_6)\wedge\mathbf{X}(s_7,s_5)\right]\mathbf{U}$$

$$\left[\bigvee_{s_5,s_6\in\Sigma}\bigvee_{t\in\Sigma\cup Q}(s_5,s_6)\wedge\mathbf{X}(\$,s_5)\wedge\mathbf{X}^2(t,\$)\right]\Big)\Big\}\Big]\Big)\Big\}$$

$$\varphi_R^+=\Big\{\bigvee_{s_1\in\Sigma}\bigvee_{q\in Q}\Big((q,\$)\wedge\mathbf{X}(s_1,\delta_2(q,s_1))\wedge$$

$$\mathbf{X}^2\Big(\Big\{(\$,\delta_1(q,s_1))\wedge\mathbf{X}\bigvee_{t\in\Sigma\cup Q}(t,\$)\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}\bigvee_{t\in\Sigma\cup Q}((s_4,\delta_1(q,s_1))\wedge\mathbf{X}(\$,s_4)\wedge\mathbf{X}^2(t,\$))\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}(s_4,\delta_1(q,s_1))\wedge\mathbf{X}\Big(\Big[\bigvee_{s_5,s_6,s_7\in\Sigma}(s_5,s_6)\wedge\mathbf{X}(s_7,s_5)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_5,s_6,s_7\in\Sigma}\bigvee_{t\in\Sigma\cup Q}(s_5,s_6)\wedge\mathbf{X}(\$,s_5)\wedge\mathbf{X}^2(t,\$)\Big]\Big)\Big\}\Big)\Big\}\vee$$

$$\Big\{\bigvee_{s_1,s_2\in\Sigma}\bigvee_{q\in Q}\Big((s_1,\$)\wedge\mathbf{X}(q,s_1)\wedge\mathbf{X}^2(s_2,\delta_2(q,s_2))\wedge$$

$$\mathbf{X}^3\Big(\Big\{(\$,\delta_1(q,s_2))\wedge\mathbf{X}\bigvee_{t\in\Sigma\cup Q}(t,\$)\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}\bigvee_{t\in\Sigma\cup Q}((s_4,\delta_1(q,s_2))\wedge\mathbf{X}(\$,s_4)\wedge\mathbf{X}^2(t,\$))\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}(s_4,\delta_1(q,s_2))\wedge\mathbf{X}\Big(\Big[\bigvee_{s_5,s_6,s_7\in\Sigma}(s_5,s_6)\wedge\mathbf{X}(s_7,s_5)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_5,s_6\in\Sigma}\bigvee_{t\in\Sigma\cup Q}(s_5,s_6)\wedge\mathbf{X}(\$,s_5)\wedge\mathbf{X}^2(t,\$)\Big]\Big)\Big\}\Big)\Big\}\vee$$

$$\Big\{\bigvee_{s,s'\in\Sigma}((s,\$)\wedge\mathbf{X}(s',s))\wedge$$

$$\mathbf{X}\Big(\Big[\bigvee_{s_1,s_2,s_3\in\Sigma}((s_1,s_2)\wedge\mathbf{X}(s_3,s_1))\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_1,s_2,s_3\in\Sigma}\bigvee_{q\in Q}\Big((s_1,s_2)\wedge\mathbf{X}(q,s_1)\wedge\mathbf{X}^2(s_3,\delta_2(q,s_3))\wedge$$

$$\mathbf{X}^3\Big(\Big\{(\$,\delta_1(q,s_3))\wedge\mathbf{X}\bigvee_{t\in\Sigma\cup Q}(t,\$)\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}\bigvee_{t\in\Sigma\cup Q}((s_4,\delta_1(q,s_3))\wedge\mathbf{X}(\$,s_4)\wedge\mathbf{X}^2(t,\$))\Big\}\vee$$

$$\Big\{\bigvee_{s_4\in\Sigma}(s_4,\delta_1(q,s_3))\wedge\mathbf{X}\Big(\Big[\bigvee_{s_5,s_6,s_7\in\Sigma}(s_5,s_6)\wedge\mathbf{X}(s_7,s_5)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_5,s_6\in\Sigma}\bigvee_{t\in\Sigma\cup Q}(s_5,s_6)\wedge\mathbf{X}(\$,s_5)\wedge\mathbf{X}^2(t,\$)\Big]\Big)\Big\}\Big)\Big]\Big)\Big\}$$

$$\varphi_H^+=\bigvee_{s_1\in\Sigma\cup\{\text{halt}\}}(s_1,\$)\wedge$$

$$\Big[\bigwedge_{s_1\in\Sigma\cup\{\text{halt},\$\}}\bigwedge_{s_2\in\Sigma\cup\{\text{halt},\$\}}\Big((s_1,s_2)\rightarrow\bigvee_{s_3\in\Sigma\cup\{\text{halt},\$\}}(s_3,s_1)\Big)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_4\in\Sigma\cup Q}(s_4,\$)\Big]\wedge$$

$$\mathbf{X}\Big(\neg\bigvee_{s_5\in\Sigma\cup Q}(s_5,\$)\mathbf{U}\Big(\bigvee_{s_6\in\Sigma\cup\{\$\}}(s_6,\text{halt})\Big)\Big)\wedge$$

$$\Big[\neg\bigvee_{s_7\in\Sigma\cup\{\$\}}(\text{halt},s_7)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_7\in\Sigma\cup\{\$\}}(\text{halt},s_7)\wedge\Big(\neg\bigvee_{s_7\in\Sigma\cup\{\$\}}(\text{halt},s_7)\mathbf{U}\bigvee_{s_8\in\Sigma\cup Q}(s_8,\$)\Big)\Big]$$

$$\varphi_{LM}^+=\Big\{\bigvee_{s_1\in\Sigma}\bigvee_{q\in Q}\Big((s_1,\$)\wedge\mathbf{X}(q,\delta_1(q,B))\wedge\mathbf{X}^2(\$,s_1)\wedge$$

$$\mathbf{X}^3\bigvee_{t_1\in\Sigma\cup Q}(t_1,\delta_2(q,B))\wedge\mathbf{X}^4\bigvee_{t_2\in\Sigma\cup Q\cup\{\$\}}(t_2,\$)\Big\}\vee$$

$$\Big\{\bigvee_{s_1\in\Sigma}\bigvee_{q\in Q}\Big((s_1,\$)\wedge$$

$$\mathbf{X}\Big(\Big[\bigvee_{s_2,s_3,s_4\in\Sigma}(s_2,s_3)\wedge\mathbf{X}(s_4,s_2)\Big]\mathbf{U}$$

$$\Big[\bigvee_{s_5,s_6\in\Sigma}\Big((s_5,s_6)\wedge\mathbf{X}(q,\delta_1(q,B))\wedge\mathbf{X}^2(\$,s_5)\wedge$$

$$\mathbf{X}^3\bigvee_{t_1\in\Sigma\cup Q}(t_1,\delta_2(q,B))\wedge\mathbf{X}^4\bigvee_{t_2\in\Sigma\cup Q\cup\{\$\}}(t_2,\$)\Big)\Big]\Big)\Big\}$$

$$\varphi_{RM}^+=\bigvee_{s_1\in\Sigma}\Big((s_1,\$)\wedge\Big($$

$$\Big\{\mathbf{X}\bigvee_{q\in Q}(q,s_1)\wedge\mathbf{X}^2(\$,\delta_1(q,B))\wedge\mathbf{X}^3\bigvee_{t_1\in Q\cup\Sigma}(t_1,\delta_2(q,B))\wedge$$

$$\mathbf{X}^4\bigvee_{t_2\in Q\cup\Sigma\cup\{\$\}}(t_2,\$)\Big\}\vee$$

$$\Big\{\mathbf{X}\Big(\Big[\bigvee_{s_2,s_3,s_4\in\Sigma}((s_2,s_3)\wedge\mathbf{X}(s_4,s_2))\Big]\mathbf{U}$$

$$\Big[\bigvee_{q\in Q}\bigvee_{s_5,s_6\in\Sigma}\Big((s_5,s_6)\wedge\mathbf{X}(q,s_5)\wedge\mathbf{X}^2(\$,\delta_1(q,B))\wedge$$

$$\mathbf{X}^3\bigvee_{t_1\in Q\cup\Sigma}(t_1,\delta_2(q,B))\wedge\mathbf{X}^4\bigvee_{t_2\in Q\cup\Sigma\cup\{\$\}}(t_2,\$)\Big)\Big]\Big\}\Big)\Big)$$

Appendix C

Definition of Δ_2^p and the polynomial hierarchy

The polynomial hierarchy PH is a subclass of $PSPACE$ that contains NP. It has an infinite number of subclasses (levels), each with complete problems. These subclasses are all conjectured to be different, but it is not yet known [AB09]. Δ_2^p is one of these subclasses. Here we define the polynomial hierarchy PH superficially, following [Sto76, AB09] by using oracle Turing machines (OTMs). An OTM $M(b)$ has the same structure as a normal Turing machine, but is augmented with an oracle O_b for some computational problem b. This means that $M(b)$ has an extra oracle tape on which it can write symbols. It then has an operation to decide whether the content of the oracle tape is a positive instance of the problem b. This operation can be used several times and on different contents of the oracle tape. By A^B we denote the class of problems that can be solved by a Turing machine $M(O_b)$ in class A (e.g. P, NP, $coNP$) with access to an oracle O_b for a complete problem b of the class B. Now the polynomial hierarchy is defined by the classes $\Pi_i^p, \Delta_i^p, \Sigma_i^p$ for $i \geq 0$ where

$$\Pi_0^p = \Delta_0^p = \Sigma_0^p = P$$

and such that for $i \geq 0$

$$\Delta_{i+1}^p = P^{\Sigma_i^p}$$

$$\Sigma_{i+1}^p = NP^{\Sigma_i^p}$$

$$\Pi_{i+1}^p = coNP^{\Sigma_i^p}$$

The polynomial hierarchy is defined by $PH = \bigcup_{i=0}^{\infty} \Sigma_i^p$.

In particular, we have from the definitions above that $\Delta_2^P = P^{\Sigma_1^p} = P^{NP^{\Sigma_0^p}} = P^{NP^P} = P^{NP}$. For a more thorough introduction to the polynomial hierarchy we refer to [Sto76, AB09].

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AHK97] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *FOCS*, pages 100–109, 1997.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[ÅW09] Thomas Ågotnes and Dirk Walther. A logic of strategic ability under bounded memory. *Journal of Logic, Language and Information*, 18(1):55–77, 2009.

[BDJ10] Nils Bulling, Jürgen Dix, and Wojciech Jamroga. Model checking logics of strategic ability: Complexity. In Mehdi Dastani, Koen V. Hindriks, and John-Jules Ch. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–158. Springer, 2010.

[BGJ12] Nils Bulling, Valentin Goranko, and Wojciech Jamroga. *Logics for Reasoning about Strategic Abilities in Multi-Player Games*. 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[BLLM09] Thomas Brihaye, Arnaud Da Costa Lopes, François Laroussinie, and Nicolas Markey. Atl with strategy contexts and bounded memory. In *LFCS*, pages 92–106, 2009.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

[CES86] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[CGP01] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, 2001.

[CL07] Taolue Chen and Jian Lu. Probabilistic alternating-time temporal logic and model checking algorithm. In *FSKD (2)*, pages 35–39, 2007.

[dAH00] Luca de Alfaro and Thomas A. Henzinger. Concurrent omega-regular games. In *LICS*, pages 141–154, 2000.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *LICS*, pages 279–290, 2001.

[DJW97] Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110, 1997.

[DT11] Catalin Dima and Ferucio Laurentiu Tiplea. Model-checking atl under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377, 1991.

[FS05] Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *LICS*, pages 321–330, 2005.

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65, 1982.

[GK07] Erich Grädel and Łukasz Kaiser. What kind of memory is needed to win infinitary muller games? In Johan van Benthem, Benedikt Löwe, and Dov Gabbay, editors, *Interactive Logic*, volume 1 of *Texts in Logic and Games*, pages 89–116. Amsterdam University Press, 2007.

[GS51] David Gale and F. M. Stewart. Infinite games with complete information. *Bulletin of the American Mathematical Society*, 57(3):175–176, 1951.

[HMU03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley, 2003.

[JÅ07] Wojciech Jamroga and Thomas Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.

[Jam08] Wojciech Jamroga. Easy yet hard: Model checking strategies of agents. In *CLIMA*, pages 1–12, 2008.

[JD08] Wojciech Jamroga and Jürgen Dix. Model checking abilities of agents: A closer look. *Theory Comput. Syst.*, 42(3):366–410, 2008.

[LMO08] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. On the expressiveness and complexity of atl. *CoRR*, abs/0804.2435, 2008.

[Mar75] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, September 1975.

[Maz01] René Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, pages 23–42, 2001.

[Mea55] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[Nas50] John F. Nash, Jr. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, January 1950.

[NRTV07] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[PR79] Gary L. Peterson and John H. Reif. Multiple-person alternation. In *FOCS*, pages 348–363, 1979.

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[PR90] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS*, pages 746–757, 1990.

[RCDH07] Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3(3), 2007.

[Rei84] John H. Reif. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci.*, 29(2):274–301, 1984.

[Sch04] Pierre-Yves Schobbens. Alternating-time logic with imperfect recall. *Electr. Notes Theor. Comput. Sci.*, 85(2):82–93, 2004.

[Sch08] Sven Schewe. *Synthesis of distributed systems.* PhD thesis, 2008.

[SLB09] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations.* Cambridge University Press, 2009.

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13, 1995.

[Tho96] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996.

[Tho02] Wolfgang Thomas. Infinite games and verification (extended abstract of a tutorial). In *CAV*, pages 58–64, 2002.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.