A LOGICAL STUDY OF STRATEGIES IN GAMES

By

Sunil Easaw Simon

The Institute of Mathematical Sciences, Chennai.

A thesis submitted to the
Board of Studies in Mathematical Sciences

In partial fulfillment of the requirements

For the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

January 2010

Homi Bhabha National Institute

Recommendations of the Viva Voce Board

As members of the Viva Voce Board, we recommend that the dissertation prepared by **Sunil Easaw Simon** entitled "A Logical Study of Strategies in Games" may be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

---------- **Date :**

Chairman : R. Ramanujam

---------- **Date :**

Convener : R. Ramanujam

---------- **Date :**

Member : Anil Seth

---------- **Date :**

Member : Kamal Lodaya

---------- **Date :**

Member : Meena Mahajan

---------- **Date :**

Member : K. Narayan Kumar

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to HBNI.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it may be accepted as fulfilling the dissertation requirement.

---------- **Date :**

Guide : R. Ramanujam

DECLARATION

I, hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and the work has not been submitted earlier as a whole or in part for a degree/diploma at this or any other Institution or University.

Sunil Easaw Simon

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Dr. R. Ramanujam for his wonderful guidance and support all through my stay at Matscience. It has taught me a lot over the past six years. His insight into games and logic is reflected in every aspect of this thesis and has brought focus and clarity to this text. His enthusiasm, ability to come up with new ideas in tackling problems and above all his immense potential to juggle with so many duties and all this with a smile, will always remain a source of inspiration to me.

The computer science groups at IMSc and CMI have been a great source of academic enrichment for me. All that I have learnt of theoretical computer science, I owe it to them. I am deeply indebted to each one of them - V. Arvind, Kamal Lodaya, Meena Mahajan, R. Ramanujam, Venkatesh Raman, C.R. Subramanian; Madhavan Mukund, K. Narayan Kumar and K.V. Subrahmanyam.

My special thanks to Soumya Paul, my colleague with whom part of the work reported here was done in collaboration with. It was an absolute pleasure to work with Soumya and his never ending enthusiasm for new ideas has always made our research discussions a fun experience. Many thanks to S.P. Suresh, who has helped me so often in both technical and non-technical matters.

I would like to thank the Director, Prof. R. Balasubramanian, for having enabled my stay at this institute during which all the work presented here has been carried out. I also thank the institute for providing a wonderful environment to pursue research. Matscience has also been instrumental in providing generous support for my various academic visits. Thanks also to the library and administrative staff of the institute, whose efforts have made my stay here so fruitful and comfortable.

Being at Matscience wouldn't have been the same without the company and camaraderie of my academic colleagues and friends here. I thank each one of them for having made my stay at Matscience so memorable. All the good times that I have had with T. Muthukumar and Piyush P. Kurur during my early days at Matscience are still vivid in my mind. My affectionate thanks to Jayalal Sarma and Ved Prakash Gupta, for being ever willing to help me in myriad ways. Thanks to N. Narayanan whom I could always bank on to clear all my Latex related doubts. He has graciously and patiently given me the benefits of his technical expertise.

I am indebted to my family in ways far more than I can express, especially to my nieces Rea and Amy; their vivacious sparkle and spontaneous reasoning have always energised me.

Abstract

The main theme of this thesis is reasoning about strategies in games in a logical framework. Logical analyses of games typically consider players' strategies as atomic objects and the reasoning is about existence of strategies, rather than about strategies themselves. This works well with the underlying assumption that players are rational and possess unbounded computational ability. However, in many practical situations players have limited computational resources. Thus a prescriptive theory which provides advice to players needs to view strategies as relations constraining players' moves rather than view them as complete functions. The central idea is to formulate the notion of composite strategies which are constructed in a structural manner and to show how explicit reasoning of strategies can be achieved.

The first part of the thesis looks at a logical analysis of strategies. We start this study by defining the notion of structurally specified strategies in the framework of unbounded duration games on graphs. This enables us to reason about how a player's strategy may depend on assumptions about the opponent's strategy. Such specifications give rise to partially specified bounded memory strategies. We consider a simple modal logic to reason about such structured strategies. We present a complete axiomatization of this logic and show that the truth checking problem of the logic is decidable.

We then look at how structurally specified strategies can be adapted to the case where the game itself has compositional structure. In this setting we suggest that rather than performing strategic reasoning on the composite game, one needs to compose game-strategy pairs. The advantage of imposing structure not merely on games or on strategies but on game-strategy pairs, is that we can speak of a composite game g followed by g' whereby if the opponent played a strategy π in g, the player responds with σ in g' to ensure a certain outcome. In the presence of iteration, a player has significant ability to strategize by taking into account the explicit structure of games. We consider a propositional dynamic logic whose programs are regular expressions over such game-strategy pairs and present a complete axiomatization of the logic. We also show that the satisfiability problem of the logic is decidable.

In the second part of the thesis, we look at an algorithmic analysis of games. We propose evaluation automata as a convenient finite state model to present the preference orderings of players in infinite games. We look at the classical solution concept of Nash equilibrium in terms of functional strategies and show that an

equilibrium profile always exists in infinite duration games on (finite) arenas where the preference orderings of players are specified in terms of evaluation automata. We also show that the best response verification question is decidable with respect to strategy specifications and that synthesizing a best response strategy is possible.

All the analysis mentioned above is carried out for games of perfect information. We finally investigate multi-player games of imperfect information. It follows from the result of Peterson and Reif (1979) that in general the verification question which asks whether a subset of players have a distributed winning strategy is undecidable in these games. The crucial element which yields undecidability is the fact that players are not allowed to communicate with each other. We propose a framework to model games of imperfect information where communication is explicitly represented. Here a player's information partition is generated in a structural manner rather than being presented as part of the game formalism. We show that for the subclass of games where communication is restricted to public announcements the verification question is decidable. We also look at the non-zero sum version where players have preference orderings over outcomes. In this setting we show that best response computation can be performed and that one can verify whether a strategy profile constitutes an equilibrium can be carried out.

Contents

List of Figures

List of Publications/Reports

[1] R. Ramanujam and S. Simon. Structured strategies in games on graphs. In *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 567–587. Amsterdam University Press, 2007.

[2] R. Ramanujam and S. Simon. A logical structure for strategies. In *Logic and the Foundations of Game and Decision Theory (LOFT 7)*, volume 3 of *Texts in Logic and Games*, pages 183–208. Amsterdam University Press, 2008. An initial version appeared in the Proceedings of the 7th Conference on Logic and the Foundations of Game and Decision Theory (LOFT06) under the title *Axioms for Composite Strategies*, pages 189-198.

[3] R. Ramanujam and S. Simon. Dynamic logic on games with structured strategies. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, pages 49–58. AAAI Press, 2008.

[4] R. Ramanujam and S. Simon. Dynamic logic on normal form games. In *Pre-Proc. Workshop on Knowledge Representation for Agents and Multi-Agent Systems (KRAMAS 2008)*, pages 140–154, 2008.

[5] R. Ramanujam and Sunil Simon. Reasoning *in* games. In *Logic, Navya-Nyaya & Applications*, volume 15 of *Studies in Logic*, pages 261–286. College Publications, 2008.

[6] S. Paul and S. Simon. Nash equilibrium in generalised Muller games. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 335–346. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2009.

Chapter 1

Introduction

The central innovation introduced by game theory is its strategic dimension. A player's environment is not neutral, and she expects that other players will try to outguess her plans. Reasoning about such expectations and strategizing one's own response accordingly constitutes the main logical challenge of game theory. Games are defined by sets of rules that specify what moves are available to each player, and according to her own preferences over the possible outcomes, every player plans her strategy. If the game is rich enough, the player has access to a wide range of strategies, and the choice of what strategy to employ in a game situation depends not only on the player's understanding of how the game can proceed from then on, but also based on his expectation of what strategies other players are following.

While this observation holds true for much of game playing, game theory largely consists of reasoning *about* games rather than reasoning *in* games. It is assumed that the entire structure of the game is laid out in front of us, and we reason from above, predicting how rational players would play, and such predictions are summarised into assertions on existence of equilibria. In an ideal world where players have unbounded computational abilities and where rationality is common knowledge, such predictions would be realistic. Players could strategize based on all possible behaviours of others and if optimal strategies exist then they will always be able to deduce these strategies. However, in reality, players are bounded memory agents having limited computational abilities. Much of the theory analysing solution concepts in games assumes that players are rational, have unbounded computational resources and talks only about the existence of stable strategy profiles.

These comments hold true even for finite duration games with perfect information. The classic example of such a game is the game of chess. Using the backward

induction algorithm, Zermelo [Zer13] argued that chess is determined, i.e. either there exists a pure strategy for one of the two players (white or black) guaranteeing that she will always win or each one of the two players has a strategy guaranteeing at least a draw. However, neither do we know which of the three alternatives is the correct one, nor a winning strategy if it exists. For games like Hex, it is known that the first player can force a win [Gal79] but nonetheless a winning strategy is not known. Theoretically a finite game like chess or hex is not very interesting since the winner can be determined in time linear in the size of the game tree using the backward induction procedure.

As is apparent, existence results are of not much help in advising players on how to play. The situation gets worse in the case of games with overlapping objectives where solution concepts in general look for equilibrium strategy profiles where none of the players gain by unilaterally deviating. In general, such games can have multiple equilibrium profiles and it is not clear which equilibrium the players would try to attain. An equilibrium selection theory was proposed by Harsanyi and Selten in [HS87] to deal with such situations. The theory models the uncertainty of each player in terms of a belief hierarchy which specifies a player's beliefs about what others play, about what they believe she and others play and so on, ad infinitum. The theory thus makes use of unbounded iteration of beliefs and it is hardly clear whether this matches in any way the reasoning done by players when they actually play a game.

And yet, as Aumann and Dreze [AD05] point out, game theory started by trying to develop a prescriptive theory for rational agents. The seminal work of von Neumann and Morgenstern envisaged game theory as constituting advice for players in game situations, so that strategies may be synthesized accordingly. While this was summarily achieved for two person zero sum games, advice functions for multiplayer games with overlapping objectives have been hard to come by. [AD05] argue that such a prescriptive game theory must account for the beliefs and expectations each player has about strategies followed by other players. The interactive element is crucial and a rational player should then play so as to maximize his utility, given how he thinks the others will play.

We suggest that any prescriptive theory which takes into account the limited computational abilities of players needs to consider strategies as partial plans rather than complete ones. Or in other words, strategies need to be considered as relations constraining players' moves, rather than functions prescribing them uniquely.

Thus rather than viewing them as atomic objects, strategies need to be viewed as structured objects built in some compositional fashion. This calls for a syntactic grammar for composition of partial strategies and it also suggests that logical languages designed to reason about composition of programs could provide valuable insight in developing a similar framework for strategies.

Logical analysis of games and strategies

Various logical formalisms have been used to reason about games and strategies, and they can broadly be classified into three branches.

Modal and dynamic logic: In the case of finite extensive form games, action indexed modal logics are well suited for logical analysis. Utilities can be coded up in terms of special propositions and the preference ordering is then induced by the implication available in the logic. Game trees themselves are taken as models of the logic. Adopting this approach, a characteristic formula for the backward induction procedure is exhibited in [Bon01]. In [Ben01, Ben02] van Benthem argues that extensive form games can be thought of as process models along with special annotations identifying player nodes. A dynamic logic framework can then be used to describe complete strategies of players as well as reasoning about outcomes that can be ensured. Instead of coding objectives of players in terms of propositions, there have been suggestions to incorporate elements of modal preference languages into the logic.

Explicit coding of complete strategies works well in the case of finite extensive form games. However, this approach is not "generic", and the description of strategies depends on the particular model under consideration. The dynamic logic formalism codes up the exact sequence of moves which form a complete strategy and this crucially relies on having a specific bound on the depth of the game tree. But then, this technique does not help in the analysis of unbounded duration games. We often come across games where players have the option to quit the game or continue playing. If they continue, they can potentially earn a better payoff but there is also the downside of losing what they have earned. These are games where the plays are finite but of unbounded duration; such games can be easily modelled as games on graphs.

Temporal logic: Various temporal logics have also been employed to reason about

games. Notable among these is the work on alternating temporal logic (ATL) [AHK02] which considers selective quantification over paths that are possible outcomes of games in which players and an environment alternate moves. ATL reasons about structured games which are games on graphs where each node is associated with a single normal form game. In the initial formulation of ATL strategies themselves could not be referred to in the logic. Various extensions of ATL where strategies are allowed to be named and referred to in the formulas of the logic are proposed in [vdHJW05] and [WvdHW07]. We will look at reasoning in games with ATL in more detail in Section 3.3.7.

Game logic: Propositional game logic [Par85], the seminal work on logical aspects of game theory, talks of existence of strategies but builds composite structure into games. [Gor03] looks at an algebraic characterisation of games and presents a complete axiomatization of identities of the basic game algebra. Pauly [Pau01] has built on this to provide interesting relationships between programs and games, and to describe coalitions to achieve desired goals. Goranko [Gor01] relates Pauly's coalition logics with work done in alternating temporal logic. In this line of work, the game itself is structurally built from atomic objects. However, the reasoning done is about existence of strategies and strategies themselves do not figure in the logical formalism.

Thus existing work on logical analysis of games tend to focus on "existence of strategies" for players. Even when strategies are considered explicitly in the logical formalism they are taken to be complete plans and mostly memoryless. To get a better understanding of the logical foundations of game theory it is necessary to look at formalisms which take into account the structure of strategies explicitly in the logical language. In van Benthem's words, strategies are the "unsung heroes" of game theory [Ben07]. A recent work in this direction is that of Ghosh [Gho08] which presents a complete axiomatisation of a logic describing both games and strategies in a dynamic logic framework, but again the assertions are about unstructured strategies.

In this thesis, we take up the issue of logical analysis of strategies in games. More precisely, we propose a syntactic structure to specify bounded memory strategies in terms of their observable properties and look at how reasoning *in* games can be effectively done with respect to strategy specifications.

Game representations and solution concepts

A natural way of presenting games is by representing the individual moves of players in an explicit manner. This is often called the extensive form game representation, where the game is represented as a tree with nodes representing game positions and edges representing the moves of players. Since the extensive form representation preserves the structure of the game, it is an ideal representation to reason about strategies which are structured in terms of its observable properties. In contrast, the normal form representation, which is widely used in game theoretic literature provides an abstract representation of games in terms of outcomes. This presentation is justified under the standard assumption made in game theory that players are rational. If players are perfectly rational and they have unbounded computational resources, then they could in principle conceive of strategies in full detail taking into account every possible move of the opponent and specifying their response. The important point being that this can be done even without playing the game. Thus the strategies of all players can be listed and the game could be specified in terms of the outcomes. The game is then played by each of the players choosing a strategy simultaneously.

However, in this approach, optimal strategies or stable strategy profiles of players need not exist. This brings us to one of the most important notions in game theory, that of mixed strategies. As opposed to deterministic strategies (also called pure strategies) which pick actions with absolute certainty, mixed strategies associate a probability distribution over the set of possible actions. We find mixed strategies often used in many of the children's games, for instance it is well known that the best strategy to follow in the game of matching pennies is for both players to pick heads or tail randomly [Str93]. The seminal result of Nash [Nas50] showed that every finite game has an equilibrium profile in mixed strategies.

Even in cases where the set of pure strategies of players is finite, the set of mixed strategies need not be. Thus to reason in games, there is the natural question of how players implement such strategies. One way is to think of players dynamically switching between pure strategies based on the observed behaviour of other players. In this context an interesting question would be to ask whether the behaviour of players stabilizes. In other words, do players eventually stop switching between strategies and stick to some pure strategy. A preliminary logical study of these issues were taken up in [PRS09a, PRS09b]. A more realistic interpretation of players implementing mixed strategies would be: players typically begin with some expec-

tation on the behaviour of other players and revise their expectations depending on what they observe in the history of the play. Such situations are extensively studied in classical game theory literature where Bayesian revision of priors is a standard technique. Incorporating expectations of players into the logical framework for strategic reasoning is an interesting and challenging task which will also model players' behaviour in a more realistic fashion. In this thesis however, we do not take up this issue.

Sub-game perfect equilibrium: Since extensive form games are represented in terms of trees, the notion of sub-games can be defined in a natural manner. Given an extensive form game T and a game position s the subtree rooted at s constitutes a sub-game of the original game T. Sub-game perfect equilibrium is a refinement of standard equilibrium notion where choices which involve any player making a move that is not credible (because it is not optimal) are eliminated. A strategy profile is said to be a sub-game perfect equilibrium [Sel65] if it represents an equilibrium profile in every sub-game of the original game. In other words, sub-game perfection looks at strategies which are compositionally constructed from strategies in the various sub-games of the original game. In the setting of finite games, sub-game perfection is justified under the trembling hand assumption that players may choose unintended strategies with negligible probability and thus all the game positions of the original game are reachable. In the context of unbounded duration non-zero sum games, it is no longer clear what the implications of trembling hand assumptions are, in fact even coming up with appropriate notions of rationality which justifies the trembling hand assumption is a challenging task. However, the equilibrium notions can be mathematically well defined by extending definitions of finite game structures. If we do not question the foundational issues involved, then the material developed in Chapter 4 suggests a way of composing strategies by taking into account sub-game structures.

Structure of the thesis

The thesis looks at the logical and algorithmic analysis of strategies. Chapter 2 provides a general introduction to extensive form and normal form games and the various well known solution concepts. We also give an overview on logical analysis of games in the literature. The first part of the thesis consists of Chapters 3 and 4 which look at the logical analysis of strategies. In Chapter 3 we introduce strategy

specifications and relate these specifications to partially defined bounded memory strategies. A logic for reasoning about such structured strategies with respect to a single game is defined. A complete axiomatization of the logic is presented and the truth checking problem is shown to be decidable. Chapter 4 looks at how logical analysis of strategies can be appropriately adapted in the case when the game itself is compositional. We define a logic which explicitly takes into account the compositional structure of games for strategic reasoning. We give a complete axiomatization of the logic and show that the satisfiability problem for the logic is decidable. Chapters 5 and 6 constitute the second part of the thesis which looks at the algorithmic analysis of strategies in games. In Chapter 5, we study the classical solution concept of Nash equilibrium with respect to non-zero sum infinite games in terms of functional strategies. We also look at how strategy specifications help in the algorithmic analysis of infinite non-zero sum games. In Chapter 6, we look at games of imperfect information and provide a model where communication between players is explicitly represented. We show that the best response computation can be effectively carried out in this setting. Chapter 7 contains concluding remarks and comments on future work.

Chapter 2

Preliminaries

In this chapter we provide a general introduction to the two main forms of game representation: the extensive form and the normal form representation. We introduce well known solution concepts associated with games. We also give an overview on logical analysis of games.

Throughout the thesis, we will be working in the setting of non-cooperative games. In Chapters 2,3,4 and 5, we will be dealing with games of perfect information. We therefore restrict our attention to two player games for convenience of presentation. The techniques developed can be extended to multi-player games as well. We do not analyze the effect of players forming coalitions even though it constitutes a very interesting branch of game theory. We do however look at imperfect information in Chapter 6 and analyse such games explicitly in terms of multiple players.

2.1 Extensive form games

Extensive form games are a natural model for representing finite games in an explicit manner. In this model, the game is represented as a finite tree where the nodes of the tree corresponds to the game positions and edges correspond to moves of players. The leaf nodes are labelled with payoffs obtained by players. We present the formal definition below.

Let N denote the set of players, we use i to range over this set. For technical convenience, we restrict our attention to two player games, i.e. we take $N = \{1, 2\}$. We often use the notation i and $\overline{i}$ to denote the players where $\overline{i} = 2$ when $i = 1$ and $\overline{i} = 1$ when $i = 2$. Let Σ be a finite set of action symbols representing moves of

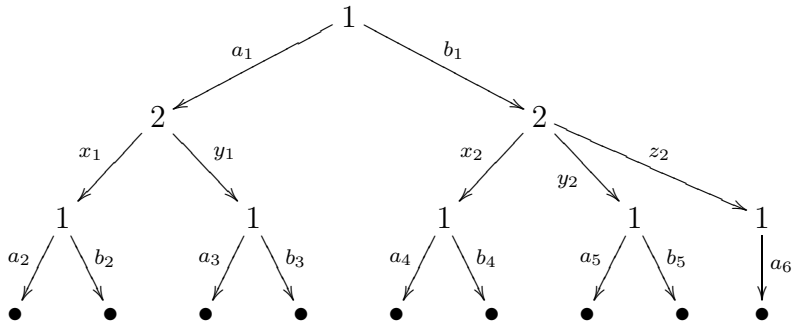


Figure 2.1: Finite extensive form game.

players, we let a, b range over Σ. For a set X and a finite sequence $\rho = x_1 x_2 \dots x_m \in X^*$, let $last(\rho) = x_m$ denote the last element in this sequence.

2.1.1 Game trees

Let $\mathbb{T} = (S, \Rightarrow, s_0)$ be a tree rooted at s_0 on the set of vertices S and $\Rightarrow : (S \times \Sigma) \to S$ is a *partial* function specifying the edges of the tree. The tree $\mathbb{T}$ is said to be finite if S is a finite set. For a node $s \in S$, let $\vec{s} = \{s' \in S \mid s \stackrel{a}{\Rightarrow} s' \text{ for some } a \in \Sigma\}$. A node s is called a leaf node (or terminal node) if $\vec{s} = \emptyset$. Let $frontier(\mathbb{T})$ denote the set of all leaf nodes of $\mathbb{T}$.

A finite extensive form game tree is a pair $T = (\mathbb{T}, \widehat{\lambda})$ where $\mathbb{T} = (S, \Rightarrow, s_0)$ is a finite tree. The set S denotes the set of game positions with s_0 being the initial game position. The edge function $\Rightarrow$ specifies the moves enabled at a game position and the turn function $\widehat{\lambda} : S \to N$ associates each game position with a player. Technically, we need player labelling only at the non-leaf nodes. However, for the sake of uniform presentation, we do not distinguish between leaf nodes and non-leaf nodes as far as player labelling is concerned. For $i \in N$, let $S^i = \{s \mid \widehat{\lambda}(s) = i\}$.

Figure 2.1 shows an example of a finite extensive form game tree. The nodes are labelled with the player labels and edges represent actions enabled for players.

A play in the game T starts by placing a token on s_0 and proceeds as follows: at any stage if the token is at a position s and $\widehat{\lambda}(s) = i$ then player i picks an action which is enabled for her at s, and the token is moved to s' where $s \stackrel{a}{\Rightarrow} s'$. Formally a play in T is simply a path $\rho : s_0 a_0 s_1 \cdots a_{k-1} s_k$ in $\mathbb{T}$ such that for all $0 \leq j < k$ $s_j \stackrel{a_j}{\Rightarrow} s_{j+1}$ and $s_k \in frontier(T)$. Note that each leaf node t denotes a play of the game which is the unique path from the root node s_0 to t. Let $Plays(T)$ denote the set of all plays in the game tree T.

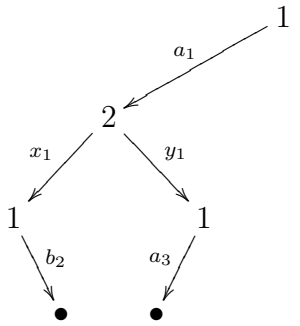


Figure 2.2: Strategy of player 1.

2.1.2 Strategies

A strategy for player i is a function μ^i which specifies a move at every game position of the player, i.e. $\mu^i : S^i \to \Sigma$. For $i \in N$, we use the notation μ^i to denote strategies of player i and $\tau^{\overline{i}}$ to denote strategies of player $\overline{i}$. By abuse of notation, we will drop the superscripts when the context is clear and follow the convention that μ represents strategies of player i and τ represents strategies of $\overline{i}$. A strategy μ can also be viewed as a subtree of T where for each player i node, there is a unique outgoing edge and for nodes belonging to player $\overline{i}$, every enabled move is included. Formally we define the strategy tree as follows:

Definition 2.1.1 *For $i \in N$ and a player i strategy $\mu : S^i \to \Sigma$ the strategy tree $T_\mu = (S_\mu, \Rightarrow_\mu, s_0, \widehat{\lambda}_\mu)$ associated with μ is the least subtree of T satisfying the following property:*

- $s_0 \in S_\mu$
- *For any node $s \in S_\mu$,*
  - *if $\widehat{\lambda}(s) = i$ then there exists a unique $s' \in S_\mu$ and action a such that $s \overset{a}{\Rightarrow}_\mu s'$.*
  - *if $\widehat{\lambda}(s) \neq i$ then for all s' such that $s \overset{a}{\Rightarrow} s'$, we have $s \overset{a}{\Rightarrow}_\mu s'$.*

Figure 2.2 shows a strategy tree for player 1 in the finite extensive form game tree given in Figure 2.1.

Let $\Omega^i(T)$ denote the set of all strategies for player i in the extensive form game tree T. A play $\rho : s_0 a_0 s_1 \cdots a_{k-1} s_k$ is said to be consistent with μ if for all

$j : 0 \leq j < k$ we have $s_j \in S^i$ implies $\mu(s_j) = a_j$. A strategy profile (μ, τ) consists of a pair of strategies, one for each player.

A play ρ is consistent with a strategy profile (μ, τ) if ρ is consistent with μ and τ. It is easy to check that given a strategy profile (μ, τ), there exists a unique play in T which is consistent with (μ, τ). This can be thought of as the play generated by (μ, τ). We denote this play by $\rho_{(\mu,\tau)}$.

2.1.3 Objectives of players

The extensive form game tree merely defines the rules of the game, or how the game progresses and terminates. More interesting are the objectives of players which specify their preferences over outcomes of the game. This is captured by associating a utility (or payoff) function $\mathbf{u}_i : \mathit{frontier}(T) \rightarrow \mathbb{V}$ with each player $i \in N$. Intuitively, for each play ρ in the game tree T terminating at some frontier node, the utility function of player i associates with ρ a value from the set $\mathbb{V}$. The payoff set $\mathbb{V}$ needs to be an ordered set, and traditionally it is taken to be a linearly ordered set. The utility function then inherits the underlying ordering of $\mathbb{V}$ which in turn induces a preference ordering on the set of plays. In what follows, we use subsets of natural numbers as the payoff set and the exact set under consideration will be explicitly mentioned. We assume the standard ordering on natural numbers.

A game is then specified by a pair $G = (T, \{\mathbf{u}_i\}_{i \in N}, \mathbb{V})$ where T is the extensive form game tree, $\mathbf{u}_i$ is the utility function of player $i \in N$ and $\mathbb{V}$ is the payoff set.

Zero sum objectives

In the case of two player games, the notion of "winning" arise in a natural way, where the objectives of players are strictly competitive. This is captured by associating each player $i \in \{1, 2\}$ with a utility function $\mathbf{u}_i : \mathit{frontier}(T) \rightarrow \{0, 1\}$ which satisfies the condition:

(Z1) For all $t \in \mathit{frontier}(T)$ and for all $i \in \{1, 2\}$, $\mathbf{u}_i(t) = 1$ iff $\mathbf{u}_{\overline{i}}(t) = 0$.

It should be noted that in the literature, the range of the utility function for two player zero sum games is usually taken to be $\{-1, 1\}$. The payoffs can always be scaled to lie in the set $\{0, 1\}$ and we choose this set merely for convenience.

A play $\rho \in \mathit{Plays}(T)$ is said to be winning for player i iff $\mathbf{u}_i(\mathit{last}(\rho)) = 1$. Due to the above restriction on the utility function, this also implies that for a play ρ which is winning for player i, $\mathbf{u}_{\overline{i}}(\mathit{last}(\rho)) = 0$. In other words, ρ is losing for player

$\overline{i}$. The utility function $\mathbf{u}_i$ induces a preference ordering $\preceq^i \subseteq Plays(T) \times Plays(T)$ in the following manner. For ρ, $\rho' \in Plays(T)$ we have

- $\rho \preceq^i \rho'$ iff $\mathbf{u}_i(last(\rho)) \leq \mathbf{u}_i(last(\rho'))$.

Note that according to the definition, $\preceq^i$ for each player i is a reflexive, transitive and complete binary relation. In other words $\preceq^i$ is a total preorder on $Plays(T)$.

A strategy μ of player i is a winning strategy if for all paths $\rho \in T_\mu$, $\mathbf{u}_i(last(\rho)) = 1$. Player i wins the game G if there exists a winning strategy for i in the game G. We say a game G is determined if one of the players win G.

Relevant problems: In the setting of two player zero sum games, the questions of interest include: Given a game $G = (T, \{\mathbf{u}_i\}_{i \in N})$,

1. is G determined?
2. given i, determine if player i wins G and if so, compute the winning strategy.

The first question concerning determinacy was settled by the following result often referred to as the Gale-Stewart theorem.

Theorem 2.1.2 ([GS53]) *Every finite two player zero sum game is determined.*

The second question deals with the algorithmic issue of determining the winner of a game and constructing the winning strategy. The backward induction algorithm due to Zermelo ([Zer13]) provides a solution to this question.

Backward induction algorithm[Jon80]: The procedure $BI(G, i)$ takes as input a game $G = (T, \{\mathbf{u}_i\}_{i \in \{1,2\}})$ and a player $i \in \{1, 2\}$. It decides whether player i has a winning strategy in G and if so, computes the winning strategy. The algorithm proceeds as follows: Initially all nodes are unlabelled.

- All leaf nodes t are labelled with $\mathbf{u}_i(t)$.
- Repeat the following steps till the root node s_0 is labelled: Choose a non-leaf node s which is not labelled and all of whose successors are labelled,
  - If $\widehat{\lambda}(s) = i$ and there exists s' such that $s \stackrel{a}{\Rightarrow} s'$ where s' is labelled 1 then label s with 1 and mark the edge $s \stackrel{a}{\Rightarrow} s'$.
  - If $\widehat{\lambda}(s) = \overline{i}$ and every successor s' is labelled 1 then label s with 1.

The correctness of the procedure is asserted by the following proposition which can be easily shown by an inductive argument.

Proposition 2.1.3 *Given a game $G = (T, \{\mathbf{u}_i\}_{i\in\{1,2\}})$ and $i \in \{1, 2\}$, player i has a winning strategy in G iff the root node s_0 is labelled with 1 by the backward induction procedure $BI(G, i)$.*

The subtree of T constituted by choosing the marked edges for all game positions of player i is the corresponding winning strategy.

Overlapping objectives

The first step in generalising from zero sum objectives would be to drop the requirement that utilities of the players be antagonistic. That is, each player is associated with a "win-loss" (or binary) objective which is specified by the utility function $\mathbf{u}_i : \mathit{frontier}(T) \to \{0, 1\}$. However, the utility function $\mathbf{u}_i$ need not satisfy condition (Z1). Thus it is possible that there exists a leaf node t such that $\mathbf{u}_i(t) = \mathbf{u}_j(t)$ for players i and j with $i \neq j$. In other words, the objectives of players can possibly overlap. In general, non-zero sum games are games where the utility function is a map $\mathbf{u}_i : \mathit{frontier}(T) \to \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural numbers.

Best response and equilibrium

In the case of non-zero sum games, the role of determinacy is replaced by one of the most important concepts in game theory, that of Nash equilibrium [Nas50]. We formally define best response and Nash equilibrium below.

Definition 2.1.4 *Given a strategy τ of player $\overline{i}$, the strategy μ of player i is the best response for τ if $\forall \mu' \in \Omega^i(T)$, $\mathbf{u}_i(\rho_{(\mu',\tau)}) \leq \mathbf{u}_i(\rho_{(\mu,\tau)})$.*

Definition 2.1.5 *A strategy profile (μ, τ) constitutes a Nash equilibrium if μ is the best response for τ and τ is the best response for μ.*

Thus a profile of strategies, one for each player, is said to be in Nash equilibrium if no player gains by unilaterally deviating from his strategy. In other words, a Nash equilibrium is a strategy profile in which every player's strategy is optimal assuming that the other players use their equilibrium strategies.

For extensive form games, a more refined notion of stable strategy profile is that of sub-game perfect equilibrium [Sel65]. Observe that for an extensive form game

$T = (S, \Rightarrow, s_0, \widehat{\lambda})$, and a node $s \in S$, the subtree of T rooted at s also defines a valid extensive form game (we denote this game by T_s). A strategy profile (μ, τ) constitutes a sub-game perfect equilibrium if for all nodes $s \in S$, the profile (μ, τ) constitutes a Nash equilibrium for the game T_s.

Relevant problems: In the context of non-zero sum games, the questions of interest include: given a game $G = (T, \{\mathbf{u}_i\}_{i \in N})$,

1. given a strategy τ of player $\overline{i}$, compute the best response strategy of player i for τ.
2. determine if game G possesses a Nash equilibrium strategy profile.
3. if Nash equilibrium exists then compute the equilibrium profile.

Below we show that for every non-zero sum finite extensive form game, a Nash equilibrium strategy profile always exists. We also show that the computation of best response and equilibrium profile is algorithmically solvable.

Best response computation: Given a game $G = (T, \{\mathbf{u}_i\}_{i \in N})$ and a strategy profile τ of player $\overline{i}$, consider the strategy tree T_τ of τ. The tree T_τ satisfies the property that for all game positions of $\overline{i}$, there is a unique outgoing edge and all choices of player i nodes are preserved. In other words, only player i has any strategic choice left. The best outcome player i can achieve is to reach a terminal node t such that $\mathbf{u}_i(t) \geq \mathbf{u}_i(t')$ for all $t' \in \mathit{frontier}(T)$. Let $\rho^t_{s_0} : s_0 a_0 \ldots s_k = t$ denote the corresponding play. Consider any strategy μ^i which satisfies the condition that for all $m : 0 \leq m < k$, $\mu^i(s_m) = a_m$. For i nodes not occurring in $\rho^t_{s_0}$ the strategy is allowed to pick any enabled action. It can be easily verified that μ^i is the best response of player i for τ.

The above procedure can be implemented to run in time linear in the size of the extensive form game tree. We need to consider the tree T_τ and find a path to the leaf node with maximum utility for player i. This path can be found using a depth first search procedure which runs in linear time and the strategy can be defined from this path.

Equilibrium computation: The backward induction algorithm introduced earlier is our core technique in equilibrium computation. We modify the earlier mentioned procedure in order to deal with utilities.

The procedure $EQ(G)$ takes as input an n-player game G and produces as output a strategy profile (μ, τ). The procedure works as follows: Initially all nodes are unmarked.

- Label all leaf nodes t with the payoff tuple $(\mathbf{u}_1(t), \mathbf{u}_2(t))$, let this be denoted by $\mathbf{u}(t)$. The labelling is then extended to all nodes of the tree as follows.
- Fix an $i \in N$ and repeat the following step till the root node s_0 is labelled: Choose a non-leaf node s for which $\mathbf{u}(s)$ is not defined and all of whose successors are labelled,
  - if $\widehat{\lambda}(s) = i$ then let $s' \in \vec{s}$ be a node such that $\mathbf{u}_i(s') \geq \mathbf{u}_i(s'')$ for all other successor nodes s'' of s. Define $\mathbf{u}(s) = \mathbf{u}(s')$ and $\mu(s) = a$ where $s \stackrel{a}{\Rightarrow} s'$.
  - if $\widehat{\lambda}(s) = \overline{i}$ then let $s' \in \vec{s}$ be a node such that $\mathbf{u}_{\overline{i}}(s') \geq \mathbf{u}_{\overline{i}}(s'')$ for all other successor nodes s'' of s. Define $\mathbf{u}(s) = \mathbf{u}(s')$ and set $\tau(s) = a$ where $s \stackrel{a}{\Rightarrow} s'$.

Note that according to the procedure, for all $i \in N$, for all nodes s, if $\widehat{\lambda}(s) = i$ then $\mu(t)$ is defined. Therefore the tuple (μ, τ) generated by the procedure constitutes a valid strategy profile. The following proposition can be shown by an inductive argument.

Proposition 2.1.6 *For a game $G = (T, \{\mathbf{u}_i\}_{i \in N})$, if (μ, τ) is the strategy profile constructed by the procedure $EQ(G)$ then for all $i \in N$, μ is the best response for τ and τ is the best response for μ.*

Corollary 2.1.7 *For a game $G = (T, \{\mathbf{u}_i\}_{i \in N})$, the strategy profile (μ, τ) constructed by the procedure $EQ(G)$ constitutes a Nash equilibrium profile.*

2.2 Normal form games

In extensive form games, moves of players are explicitly presented and therefore strategies are not abstract atomic objects but have certain structure associated with them. Another commonly used representation for games is the normal form (or strategic form) representation. In contrast to the extensive form representation, strategies are presented in normal form games in an abstract manner. In this representation, strategies of players corresponds to choosing an action from the action set.

We assume that the set of actions is partitioned into action sets for each player, i.e. $\Sigma = \Sigma_1 \cup \Sigma_2$. Let $\widehat{\Sigma} = \Sigma_1 \times \Sigma_2$. A strategy profile is simply a pair of actions, one for each player. A play of the game corresponds to each player choosing an action simultaneously without knowledge of the action picked by the other player. Thus a strategy profile constitutes a play in the game. Each play is associated with a pair of utilities, denoting the payoffs for the players.

Definition 2.2.1 *Suppose $|\Sigma_1| = m$ and $|\Sigma_2| = k$, then a strategic form game can be represented as an $m \times k$ matrix A where the actions of player 1 constitute the rows of the matrix and that of player 2 the columns. The matrix entries specify the outcome of the play for each player.*

Example 2.2.2 A normal form game with $m = k = 2$ is shown in Figure 2.3. Here $\Sigma_1 = \{a_1, b_1\}$ and $\Sigma_2 = \{a_2, b_2\}$. The action profile (a_1, a_2) where player 1 chooses to play a_1 and player 2 chooses a_2, results in the utility u_1^1 for player 1 and u_2^1 for player 2.

	a_2	b_2
a_1	(u_1^1, u_2^1)	(u_1^2, u_2^2)
b_1	(u_1^3, u_2^3)	(u_1^4, u_2^4)

Figure 2.3: Normal form game

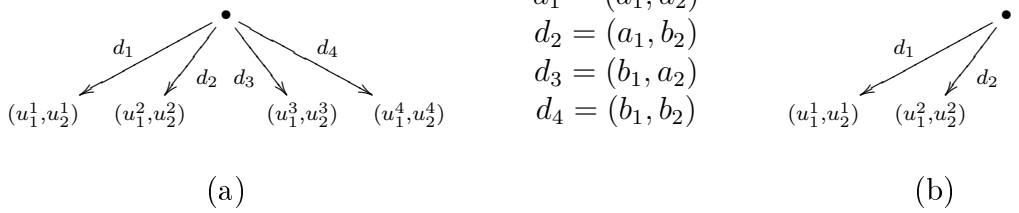
□

For normal form games, the notion of best response and equilibrium can be defined as in the case of extensive form games.

Tree representation of normal form games

A normal form game can be viewed as an extensive form game tree of depth one where the edges are labelled by pairs of actions, one for each player. Formally the game tree $T = (S, \Rightarrow, s_0, \{\mathbf{u}_i\}_{i \in N})$ where S is the set of states, s_0 is the root of the tree. The transition function $\Rightarrow : s_0 \times \widehat{\Sigma} \rightarrow (S \setminus \{s_0\})$ is a partial function also called the move function which satisfies the condition: for all $s \in S \setminus \{s_0\}$, there exists

$a \in \widehat{\Sigma}$ such that $s_0 \stackrel{a}{\Rightarrow} s$. The utility function $\mathbf{u}_i : S \to \mathbb{N}$. Let $\widehat{\Sigma}^T = \{a \in \widehat{\Sigma} \mid \exists s' \in S$ where $s_0 \stackrel{a}{\Rightarrow} s'\}$ denote the set of all strategy profiles of T.



(a) (b)

Figure 2.4: Tree representation of a normal form game

The game tree corresponding to the normal form game in Figure 2.3 is shown in Figure 2.4(a). A play is simply an edge in the tree, this corresponds to both the players picking an action. A strategy for player i is the subtree of g where for player i a unique action is chosen and for player $\overline{i}$ all the actions are taken into account. A strategy for player 1 in the game given in Figure 2.4(a) where he picks action "a_1", is shown in Figure 2.4(b).

2.3 Games on graphs

Extensive form game trees provide a convenient representation of finite games. However, we often come across games where plays are finite but the number of moves (or the depth of the game tree) is not fixed in advance. Typical examples of such games are ones in which at certain positions, players are given the option of quitting the game and walking away with a certain payoff or continue playing the game. These are usually non-zero sum games and the action of quitting the game comes with a cost tradeoff. If the player stays on and continues with the game, he can potentially get a better payoff but there is also a downside of losing what he has earned so far. If the game reaches a situation where all the players recognise that they cannot improve their payoffs any further then they can mutually agree to terminate the game.

For instance, consider the game of chess played as part of a television game show. Two players are competing against each other. However it is not a simple win-loss game. Players earn some specified amount on being able to perform a certain sequence of moves or on reaching some particular board configuration. At this point,

the players also have the option to quit the game and leave with the amount they won or to continue playing the game. If they continue, they can potentially earn more but there is also the downside of losing what they have earned. As in any game show, the payoffs are not cumulative, and they arise from a finite set of divisions. The jackpot, of course, is when a player wins the game he gets paid 1 million dollars. If the game reaches a point where the board configuration keeps repeating and the players realise that there is no way of improving their payoffs, they can mutually agree to terminate the game and disperse with what they earned so far [1].

These are games of unbounded duration and the extensive form game representation results in an infinite tree. There are various possible options to present such games in a finite fashion. One way is to specify the game structure in terms of a finite set of game rules. In this thesis, we adopt a simpler approach based on graphical game models which is to present the game in terms of a finite game arena (finite graph). The associated infinite extensive form game is obtained by the unfolding of this arena.

2.3.1 Game arena

We use games on graphs to model games of unbounded duration. We assume the existence of a special game position called *exit* to model the termination of the game and an action *quit* which corresponds to players choosing to quit the game.

Definition 2.3.1 *A game arena is a structure* $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ *where*

- W *is a finite set of game positions.*
- $\rightarrow : W \times \Sigma \rightarrow W$ *is a partial function also called the move function which satisfies the condition*
  - $\vec{exit} = \emptyset$
  - $w \xrightarrow{quit} w'$ *iff* $w' = exit$
- $w_0 \in W$ *is the initial game position.*
- $\lambda : W \rightarrow N$ *is the turn function which associates each game position with a player.*

[1] In fact one can consider any perfect information game where there is an option of increasing the stakes or quitting and the possibility of the game ending in a draw.

Let $\rightarrow(w, a) = w'$, we often denote this by $w \xrightarrow{a} w'$.

For $i \in N$, let $W^i = \{w \in W \mid \lambda(w) = i\}$. A play in $\mathcal{G}$ is either a finite path $\rho = w_0 a_0 w_1 a_1 \ldots w_k$ such that $w_k = \mathit{exit}$ or an infinite path $\rho = w_0 a_0 w_1 a_1 \ldots$ such that for all $j \geq 0$, $w_j \xrightarrow{a_j} w_{j+1}$. Let $\mathit{Plays}(\mathcal{G})$ denote the set of all plays in $\mathcal{G}$. The (infinite) extensive form game tree $T_{\mathcal{G}}$ associated with $\mathcal{G}$ is obtained by the tree unfolding of $\mathcal{G}$ which we define below.

Definition 2.3.2 *Let $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ be a game arena. The tree unfolding of $\mathcal{G}$ is the least tree structure $T_{\mathcal{G}} = (S, \Rightarrow, s_0, \widehat{\lambda})$ where $S \subseteq (W \times \Sigma)^* W$ and $\Rightarrow : S \times \Sigma \rightarrow S$ satisfies the condition:*

- $w_0 \in S$.
- *If $s = (w_0, a_0) \ldots w_k \in S$ and $w_k \xrightarrow{a} w'$ then $s' = (w_0, a_0) \ldots (w_k, a) w' \in S$ and $s \stackrel{a}{\Rightarrow} s'$.*

Further, for a node $s = (w_0, a_0) \ldots w_k \in S$, $\widehat{\lambda}(s) = \lambda(w_k)$.

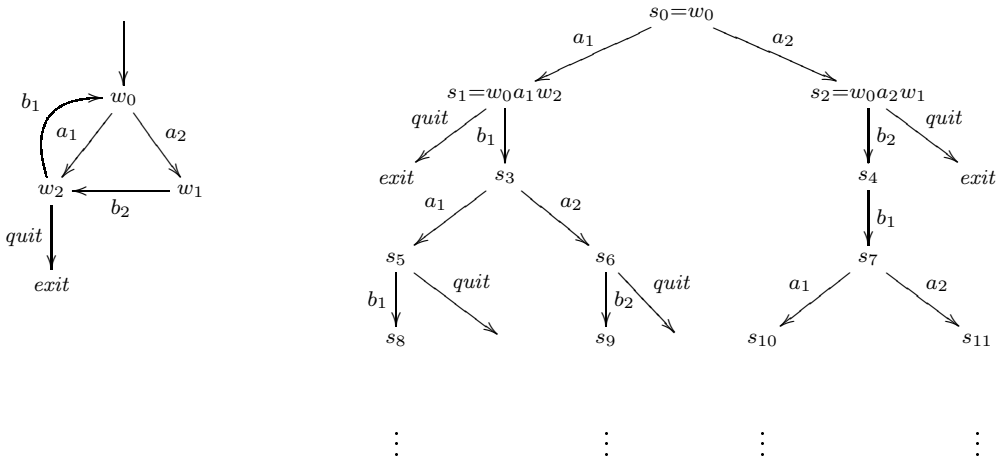


Figure 2.5: Game arena and tree unfolding

Figure 2.5 illustrates a game arena and its tree unfolding.

It is easy to see that the edge relation in $T_{\mathcal{G}}$ defines a partial function. A node $s = w_0 a_0 w_1 a_1 \ldots w_k$ in $T_{\mathcal{G}}$ denotes a finite play in the arena, with $\mathit{last}(s) = w_k$. Note that for any node s, $\mathit{last}(s) \in W$.

The objectives of players can be specified as preference orderings on the set of plays, i.e. each player is associated with an ordering $\preceq^i \subseteq \mathit{Plays}(\mathcal{G}) \times \mathit{Plays}(\mathcal{G})$.

2.3.2 Strategies

A strategy μ for player i specifies for each partial play ending in a game position of player i which action to choose. Thus μ is a map $\mu : (W \times \Sigma)^* W^i \to \Sigma$. As in the case of extensive form games (definition 2.1.1), a strategy can be viewed as a subtree of the game tree $T_{\mathcal{G}}$ where for every player i node there is a unique outgoing edge and for an $\bar{\imath}$ node, all outgoing edges are included. Since $T_{\mathcal{G}}$ is infinite, a strategy thus defined can in principle depend on the complete history of play and in general it need not be computable. For computationally bounded players it is not possible to implement or even choose to play such an arbitrarily defined strategy.

Resource limitations on strategies have been studied in the game theoretic framework. [FW94] looks at the situation where players are represented by polynomial time turing machines. This is motivated by the intuition that typically in the real world, players are equipped with powerful computers which help them optimize their strategies. In this approach the standard complexity measure notions on turing machines can be adopted to classify strategies. A weaker form of computation which is however extremely robust is represented by finite state automata. Strategies defined by finite state automata have been studied in [Ney85, AR88, Ney98]. The typical complexity measure looked at here is the size of the minimal deterministic automaton which represents the strategy. Also of interest is the notion of "program equilibrium" [Ten04] which looks at strategies which are implementable as programs and reasons about players' abilities to refer to computation done in parallel by other programs.

In this thesis we stick to the weaker form of defining resource limited strategies in terms of finite state automata. In this context, the following two types of strategies are of particular interest:

- Memoryless (positional) strategies: These are strategies for which the next move depends only on the current game position and not on the history of play. Thus the map $\mu : W^i \to \Sigma$ prescribes the same action for all partial plays ending at the same game position. That is, for all ρ, ρ' such that $last(\rho) = last(\rho')$, $\mu(\rho) = \mu(\rho')$.

- Bounded memory strategies: These are strategies where the dependence of the next move to the history of the play can be kept track of by a finite set of states. Such strategies can be represented using finite state machines. We think of these as advice automata, in the sense that they constitute an advice

for the player to consult at a game position.

Definition 2.3.3 *Let $\mathcal{G}$ be a game arena, a deterministic advice automaton for player i is a tuple $\mathcal{A} = (Q, \delta, o, q_0)$ where*

- *Q is the set of states.*
- *$\delta : Q \times W \times \Sigma \to Q$ is the deterministic transition function.*
- *$o : Q \times W^i \to \Sigma$ is the advice function.*
- *$q_0 \in Q$ is the initial state.*

For a partial play $\rho : w_0 a_0 w_1 \dots w_k$, the run of $\mathcal{A}$ on ρ is the sequence of states $q_0 q_1 \dots q_k$ such that for all j where $0 \leq j < k$, $q_{k+1} = \delta(q_k, w_j, a_j)$. The strategy μ of player i generated by $\mathcal{A}$ is defined as follows. For any partial play $\rho : w_0 a_0 w_1 \dots w_k$, let $q_0 q_1 \dots q_k$ be the run of $\mathcal{A}$ on ρ, then $\mu(\rho) = o(q_k, w_k)$. Since $\mathcal{A}$ is a deterministic automaton, it is easy to see that the strategy generated by $\mathcal{A}$ is unique.

For technical purposes, we find it convenient to define the notion of the language accepted by an advice automaton. Let μ be a strategy of player i and $T_\mu = (S_\mu, \Rightarrow_\mu, s_0)$ be the corresponding strategy tree. The run of $\mathcal{A}$ on T_μ is a Q labelled tree $T = (S_\mu, \Rightarrow_\mu, s_0, f)$, where f maps each tree node to a state in Q as follows: $f(s_0) = q_0$, and for any s_k where $s_k \overset{a}{\Rightarrow}_\mu s'_k$, we have $f(s'_k) = \delta(f(s_k), last(s_k), a_k)$.

A Q labelled tree T is accepted by $\mathcal{A}$ if for every tree node $s \in S_\mu$ where $s \in S^i$, $s \overset{a}{\Rightarrow} s'$ implies $o(f(s), last(s)) = a$. We say a strategy μ is accepted by $\mathcal{A}$ or is in the language of $\mathcal{A}$ (denoted $Lang(\mathcal{A})$) if the run of $\mathcal{A}$ on μ is accepting. For a state q and a tree node s, we often use the notation $o(q, s)$ to denote $o(q, last(s))$.

2.4 Logical analysis of games

In this section we look at logical analysis of games in the literature. We focus on modal and dynamic logics which considers the game structure to be atomic and where additional compositional structure of the game representation itself is not taken into account. Finite extensive form games are particularly suited for this type of analysis. This was suggested in [Ben02, Ben01] and has also been explored in various directions. The core idea used by the various logics adopting this style of reasoning is the following:

- Finite extensive form game tree can be viewed as models of dynamic logic.

- Strategies of players in finite extensive form games, can be encoded as programs in dynamic logic.

Here we illustrate how dynamic logic can be effectively used to reason about games and abilities of players to ensure certain outcomes. We find it instructive to first give a brief introduction to propositional dynamic logic (PDL). A more comprehensive treatment of PDL can be found in [HKT00].

Propositional dynamic logic

Dynamic logic, a logic to reason about the behaviour of composite nondeterministic programs was originally proposed by Pratt [Pra76], where it was shown how to extend modal logic by considering a separate modality for every program. Propositional reasoning about programs in terms of dynamic logic was studied by Fischer and Ladner in [FL77, FL79].

Syntax: Let P denote a countable set of propositions. The formulas of PDL are constructed using the following syntax:

$$\mathbb{P} := a \in \Sigma \mid \gamma_1; \gamma_2 \mid \gamma_1 \cup \gamma_2 \mid \gamma^* \mid \alpha?$$

$$\textbf{PDL} := p \in P \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle\gamma\rangle\alpha$$

where $\gamma \in \mathbb{P}$ and $\alpha \in \textbf{PDL}$.

Semantics: PDL formulas are interpreted over edge labelled Kripke structures $M = (W, \rightarrow, V)$ where W is the set of states, $\rightarrow \subseteq W \times \Sigma \times W$ and $V : W \rightarrow 2^P$. The semantics of composite programs is defined in terms of the relation $R^{PDL} \subseteq W \times W$ as follows.

- $R_a^{PDL} = \{(u, w) \mid u \xrightarrow{a} w\}$.
- $R_{\gamma_1;\gamma_2}^{PDL} = \{(u, w) \mid \exists v \in W \text{ where } (u, v) \in R_{\gamma_1}^{PDL} \text{and } (v, w) \in R_{\gamma_2}^{PDL}\}$.
- $R_{\gamma_1 \cup \gamma_2}^{PDL} = R_{\gamma_1}^{PDL} \cup R_{\gamma_2}^{PDL}$.
- $R_{\gamma^*}^{PDL} = \bigcup_{n \geq 0} (R_\gamma^{PDL})^n$, where $(R_\gamma^{PDL})^n$ denotes the n-fold relational composition.
- $R_{\alpha?}^{PDL} = \{(u, u) \mid M, u \models \alpha\}$.

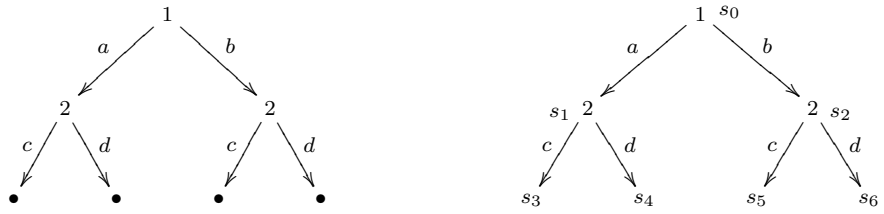
The truth of a formula α in a model M at a state u (denoted $M, u \models \alpha$) is defined inductively as follows:

- $M, u \models p$ iff $p \in V(u)$.
- $M, u \models \neg\alpha$ iff $M, u \not\models \alpha$.
- $M, u \models \alpha_1 \vee \alpha_2$ iff $M, u \models \alpha_1$ or $M, u \models \alpha_2$.
- $M, u \models \langle\gamma\rangle\alpha$ iff there exists $(u, w) \in R_\gamma^{PDL}$ such that $M, w \models \alpha$.

Reasoning about games in PDL

An extensive form game tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ can be viewed as a Kripke structure $M = (S, \Rightarrow, V)$ where $V : S \to 2^P$ is a valuation function which provides interpretation for the atomic propositions in P. The turn function $\widehat{\lambda}$ can be coded in terms of special propositions $\mathbf{turn}_i$ for each $i \in N$. It is also convenient to represent the frontier nodes of the tree in terms of a proposition **leaf**. Thus the valuation function satisfies the following condition: for all $s \in S$,

- $\forall i \in N$, $\mathbf{turn}_i \in V(s)$ iff $\widehat{\lambda}(s) = i$.
- $\mathbf{leaf}_i \in V(s)$ iff $s \in \mathit{frontier}(T)$.



(a): Game tree (b): Model

Figure 2.6: Extensive form game tree and corresponding model

Consider the game tree given in Figure 2.6(a). The model corresponding to the game tree is shown in Figure 2.6(b). Suppose the proposition p is true at states s_4 and s_5, i.e. $p \in V(s_4)$ and $p \in V(s_5)$. Consider the following PDL formula interpreted over the model given in Figure 2.6(b).

- $[(\mathbf{turn}_1?; a) \cup (\mathbf{turn}_1?; b)]\langle(\mathbf{turn}_2?; c) \cup (\mathbf{turn}_2?; d)\rangle(\mathbf{leaf} \supset p)$.

This says that no matter what move player 1 makes, player 2 has a move to ensure the outcome states satisfies p. In other words, player 2 has a "winning strategy" to ensure the outcome p.

In PDL one can talk about not just winning strategies but also about strategic response of players. Consider the formula

- $[(\mathbf{turn}_1?; a); (\mathbf{turn}_2?; d)](\mathbf{leaf} \supset p)$.

The formula asserts that if player 1 plays "a" then player 2 can respond with "b" to achieve the outcome p. In general if σ is the strategy of player 1 encoded as a PDL program and π that of player two then the following formula says that the state reached when players employ the strategy σ and π against each other satisfies the property p.

- $[(\mathbf{turn}_1?; \sigma \cup \mathbf{turn}_2?; \pi)^*]\mathbf{leaf} \supset p$.

Encoding utilities in the model: The above examples illustrate how programs can be effectively used to code strategies in finite extensive form games. More interesting would be to reason about games with respect to the utilities of players. Utilities can again be coded in terms of propositions as illustrated in [Bon02]. Let $\mathbb{V} = \{u^1, u^2, \ldots\}$ be a value set and for $i \in N$, let $\mathbf{u}_i : \mathit{frontier}(T) \rightarrow \mathbb{V}$. Without loss of generality assume that $u^1 \leq u^2 \leq \ldots$. Let $\Theta_1 = \{\theta_1^1, \theta_1^2, \ldots\}$ be a set of special propositions used to encode the utilities in the logic, i.e. θ_1^j corresponds to the utility θ_1^j. Likewise for player 2, we have the set of special propositions Θ_2. The valuation function satisfies the condition:

- For all states s, for $i \in N$, $\{\theta_i^1, \ldots, \theta_i^j\} \subseteq V(s)$ iff $\mathbf{u}_i(s) = u^j$.

The preference ordering on the rewards for each player is simply inherited from the implication available in the logic. Thus strategy comparison with respect to utilities can be expressed in the logic as follows.

- $([(\mathbf{turn}_1?; \sigma_1)^*]\mathbf{leaf} \supset \theta_i) \supset ([(\mathbf{turn}_1?; \sigma_1')^*]\mathbf{leaf} \supset \theta_i)$.

Related work: There have been various studies which build on this basic idea of using dynamic logic to reason about games. [HvdHMW03] suggests interpreting the atomic actions in PDL in terms of strategy profiles. This enables assertions to be made regarding the resulting plays in the game. The authors come up with characteristic formulas for Nash equilibrium and sub-game perfect equilibrium in this setting.

In the backward induction procedure, the idea is to lift the preference ordering over utilities to ordering over tree nodes. But as pointed out in [Ben06], when orderings are interpreted over tree nodes, it does not merely represent what players

prefer but what they expect to happen given the rationality assumptions about how other players will proceed. Instead of coding utilities directly as propositions a more elegant method would be to incorporate elements of a preference language into the logic. This was suggested in [BOR06] where the authors analyse the backward induction procedure in terms of a preference modality.

Remarks: It should be noted that the above mentioned technique of coding strategies as programs in PDL works well since we are dealing with finite extensive form games. The fact that the depth of the game tree is known in advance, is crucially used. In particular, this approach would not work in the case of unbounded duration games where the length of plays is not determined in advance. To make assertions about existence of strategies one could look at adding a fix-point operator to the logic as done in μ-calculus [Koz83]. In the case of infinite two player zero sum games on finite graphs, it is well known that winning regions of players can be expressed as μ-calculus properties (see [GTW02], Chapter 10). However, it is hardly clear how to reason about strategies in the μ-calculus framework.

Part I

Logical analysis

Chapter 3

Logical analysis of strategies

As demonstrated in the previous chapter, game theoretic analysis typically involve outcome based analysis in terms of solution concepts. However, a prescriptive theory needs to view strategies as partial functions built in a compositional manner. In this chapter we propose a programming language syntax for building partially specified strategies in a compositional framework. The framework is not dependent on any specific bound on the length of the play and is therefore suited for analysis of unbounded duration games as well.

3.1 Strategy specifications

We fix the following notations. Let $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ denote the game arena. Let $P^i = \{p^i_0, p^i_1, \ldots\}$ be a countable set of observables for $i \in N$.

We conceive of strategies as being built up from atomic ones using some grammar. The atomic case specifies, for a player, what conditions she tests for before making a move. We can associate with the game arena a set of observables for each player. One elegant method then, is to state the conditions to be checked as a past time formula of a simple tense logic over the observables. The structured strategy specifications are then built from atomic ones using connectives. We crucially use an implication of the form: "if the opponent is apparently playing a strategy π then play σ".

Below, for any countable set X, let $\mathit{Past}(X)$ be sets of formulas given by the following syntax:

$$\psi \in \mathit{Past}(X) := x \in X \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \diamond\!\!\!-\psi.$$

The past time formulas are interpreted over sequences. Intuitively, the modality

$\diamondsuit\psi$ asserts that sometime in the past the formula ψ holds. We also use $Voc(\psi)$ to denote the subset of X which is mentioned in ψ.

3.1.1 Syntax

The syntax of strategy specifications is given by:

$$Strat^i(P^i) := [\psi \mapsto a]^i \mid \sigma_1 + \sigma_2 \mid \sigma_1 \cdot \sigma_2 \mid \pi \Rightarrow \sigma_1$$

where $\pi \in Strat^{\overline{i}}(P^1 \cap P^2)$ and $\psi \in Past(P^i)$.

The idea is to use the above constructs to specify properties of strategies. For instance the interpretation of a player i specification $[p \mapsto a]^i$ where $p \in P^i$ is to choose move "a" at every player i game position where p holds. At positions where p does not hold, the strategy is allowed to choose any enabled move. $\sigma_1 + \sigma_2$ says that the strategy of player i conforms to the specification σ_1 or σ_2. The construct $\sigma_1 \cdot \sigma_2$ says that the strategy conforms to specifications σ_1 and σ_2.

The specification $\pi \Rightarrow \sigma$ says, at any node player i sticks to the specification given by σ if on the history of the play, all moves made by $\overline{i}$ conform to π. In strategies, this captures the aspect of players' actions being responses to the opponent's moves. The opponent's complete strategy may not be available, the player makes a choice taking into account the apparent behaviour of the opponent on the history of play.

Let $\Sigma = \{a_1, \ldots, a_m\}$, we also make use of the following abbreviation.

- $null^i = [True \mapsto a_1] + \cdots + [True \mapsto a_m]$.

It will be clear from the semantics (which is defined shortly) that any strategy of player i conforms to $null^i$, or in other words this is an empty specification. The empty specification is particularly useful for assertions of the form "there exists a strategy" where the property of the strategy is not of any relevance.

3.1.2 Semantics

Given any sequence $\xi = t_0 t_1 \cdots t_m$, a map $V : \{t_0, \cdots, t_m\} \to 2^X$, and an index k such that $0 \leq k \leq m$, the truth of a past formula $\psi \in Past(X)$ at k, denoted $\xi, k \models \psi$ can be defined as follows:

- $\xi, k \models p$ iff $p \in V(s_k)$.
- $\xi, k \models \neg\psi$ iff $\xi, k \not\models \psi$.

- $\xi, k \models \psi_1 \vee \psi_2$ iff $\xi, k \models \psi_1$ or $\xi, k \models \psi_2$.
- $\xi, k \models \Diamond\psi$ iff there exists a $j : 0 \leq j \leq k$ such that $\xi, j \models \psi$.

Strategy specifications are interpreted on strategy trees of $\mathcal{G}$. For this purpose, we consider the game arena $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ along with a valuation function for the observables $V : W \rightarrow 2^P$. We assume the presence of two special propositions $\mathbf{turn}_1$ and $\mathbf{turn}_2$ that specifies which player's turn it is to move, i.e. the valuation function satisfies the property

- for all $i \in N$, $\mathbf{turn}_i \in V(w)$ iff $\lambda(w) = i$.

The valuation function V is extended to the tree unfolding $T_{\mathcal{G}}$ in the obvious manner: for a node s we have $V(s) = V(last(s))$. Recall that a strategy μ of player i is a subtree of $T_{\mathcal{G}}$. Given a strategy μ of player i and a node $s \in \mu$, let $\rho^s_{s_0} : s_0 a_0 s_1 \cdots s_m = s$ be the unique path in μ from the root node s_0 to s. For a strategy specification $\sigma \in Strat^i(P^i)$, we define the notion of μ conforming to σ (denoted $\mu \models_i \sigma$) as follows:

- $\mu \models_i \sigma$ iff for all player i nodes $s \in \mu$, we have $\rho^s_{s_0}, s \models_i \sigma$.

where we define $\rho^s_{s_0}, s_j \models_i \sigma$ for any player i node s_j in $\rho^s_{s_0}$ as,

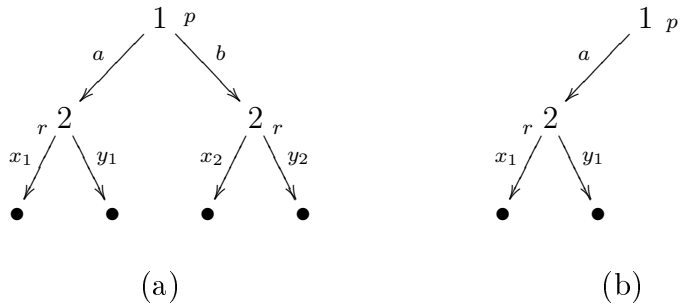
- $\rho^s_{s_0}, s_j \models_i [\psi \mapsto a]^i$ iff $\rho_s, s_j \models \psi$ implies $out_{\rho^s_{s_0}}(s_j) = a$.
- $\rho^s_{s_0}, s_j \models_i \sigma_1 + \sigma_2$ iff $\rho^s_{s_0}, s_j \models_i \sigma_1$ or $\rho^s_{s_0}, s_j \models_i \sigma_2$.
- $\rho^s_{s_0}, s_j \models_i \sigma_1 \cdot \sigma_2$ iff $\rho^s_{s_0}, s_j \models_i \sigma_1$ and $\rho^s_{s_0}, s_j \models_i \sigma_2$.
- $\rho^s_{s_0}, s_j \models_i \pi \Rightarrow \sigma_1$ iff (if for all player $\overline{i}$ nodes $s_k \in \rho^s_{s_0}$ such that $k \leq j$, $\rho^s_{s_0}, s_k \models_{\overline{i}} \pi$) then $\rho^s_{s_0}, s_j \models_i \sigma_1$.

Above, $\pi \in Strat^{\overline{i}}(P^1 \cap P^2)$, $\psi \in Past(P^i)$, and for all $i : 0 \leq i < m$, $out_{\rho^s_{s_0}}(s_i) = a_i$ and $out_{\rho^s_{s_0}}(s)$ is the unique outgoing edge in μ at s. Recall that s is a player i node and therefore by definition there is a unique outgoing edge at s.

Given a game arena $\mathcal{G}$, a player $i \in N$ and a strategy specification $\mu \in Strat^i(P^i)$ we define the tree language $\mathcal{TL}(\mathcal{G}, \sigma)$ as $\mathcal{TL}(\mathcal{G}, \sigma) = \{\mu \in \Omega^i(\mathcal{G}) \mid \mu \models_i \sigma\}$.

Remarks

Note that we do not have negation in specifications. One reason is that the specifications are partial, and hence the semantics is not immediate. If we were to consider a specification of the form $\overline{\pi} \Rightarrow \sigma$, we could interpret this as: if player has seen that opponent has violated π in the past, then play σ. This seems rather unnatural, and hence, for the present, we are content to leave negation aside. Note that we do have negation in tests in atomic specifications, and later we will embed these specifications into a modal logic (with negation on formulas).



(a) (b)

Figure 3.1: Game and strategy.

Example 3.1.1 Consider the game given in figure 3.1(a) and the specification $[p \mapsto a]^1$. The interpretation is to choose move "a" for every 1 node where proposition p holds. Suppose p holds at the root, then the strategy depicted in Figure 3.1(b) conforms to the specification $[p \mapsto a]^1$. □

According to the syntax it is possible to come up with arbitrary strategy specifications which when interpreted on a game arena could be inconsistent. For instance, consider the game given in Figure 3.2(a) and the specification $\sigma = [p \mapsto a]^1 \cdot [p \mapsto b]^1$. It is easy to see that no strategy of player 1 conforms to the specification σ. Thus a natural question would be to ask:

- given an arena $\mathcal{G}$, player $i \in N$ and a specification $\sigma \in \mathit{Strat}^i(\mathcal{G})$ is it decidable to check whether $\mathcal{TL}(\mathcal{G}, \sigma) = \emptyset$?

In what follows it will be clear that this question is decidable. The crucial fact which needs to be used is that strategy specifications classify strategies which are "regular". Thus it is possible to build a finite state automaton which recognize such strategies.

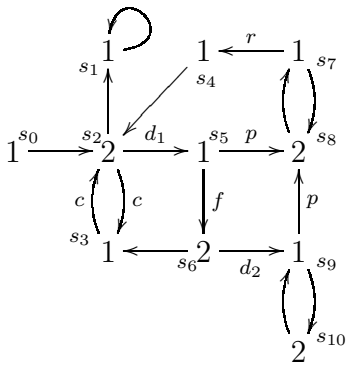


Figure 3.2: Game arena

Example 3.1.2 Consider the game shown in Fig. 3.2. Let the cycle $C_0 : s_1 \rightarrow s_1$, $C_1 : s_2 \rightarrow s_3 \rightarrow s_2$, $C_3 : s_9 \rightarrow s_{10} \rightarrow s_9$ and $C_4 : s_8 \rightarrow s_7 \rightarrow s_8$. For a player i and cycles C and C' we use the notation $C \preceq^i C'$ to denote that plays which settle down to the cycle C' is preferred by player i to plays which settle down to cycle C. Let the preference ordering of player 2 rank $\{C_0, C_4\}$ as the worst outcomes and C_3 as the best outcome with $C_1 \preceq^2 C_3$. Let the preference ordering of player 1 be given as $C_4 \preceq^1 C_3 \preceq^1 C_1 \preceq^1 C_0$. Its easy to see that equilibrium reasoning will end up in cycle C_3. However, for player 1 the utility difference between C_3 and C_4 might be negligible. So player 1 might decide to punish 2 by moving to C_4 at position s_9. Therefore it is in the common interest of both players to stick to cycle C_1.

Let $V(choice) = \{s_2, s_6\}, V(forgive) = \{s_4\}, V(decide_1) = \{s_5\}, V(decide_2) = \{s_6\}$, $V(punish) = \{s_8\}$ and $V(worst) = \{s_7\}$. A strategy specification for player 1, which forgives the first defection of player 2, and punishes if 2 has always defected so far in the play, can be written as:
$\sigma = [decide_1 \mapsto f]^1 \cdot ([choice \mapsto d]^2 \Rightarrow [decide_2 \mapsto p]^1)$.

A specification of player 1 which punishes at any defection by player 2 but tries to re-conciliate once, can be given as:
$\sigma = [decide_1 \mapsto p]^1 \cdot [(\neg \diamondsuit forgive \wedge choice) \mapsto d]^2 \Rightarrow [worst \mapsto r]$.

Player 2 might decide to cooperate if he has been punished in the past, which can be written as:
$\pi = [\diamondsuit punish \wedge choice \mapsto c]^2$. □

3.1.3 Partial strategies and advice automata

The main objective of developing a compositional syntax for strategies is to analyze strategies in terms of their properties. Strategies are thus specified as partial functions which constrains moves of players rather than prescribing unique moves. Given a game arena $\mathcal{G}$, a partial strategy ν for player i is a partial function $\nu : (W \times \Sigma)^* W^i \rightharpoonup \Sigma$, with the interpretation that if ν is not defined for some partial play $\rho \in (W \times \Sigma)^* W^i$, player i is allowed to play *any* available action there. Thus a partial strategy can equivalently be thought of as a set of complete strategies. This also motivates the definition of nondeterministic advice automata. We saw that bounded memory strategies can be represented in terms of deterministic advice automata (Definition 2.3.3). Since partial strategies represent a set of complete strategies, bounded memory partial strategies can be represented in terms of nondeterministic advice automata which we define below.

Definition 3.1.3 *For a game arena $\mathcal{G}$, a nondeterministic advice automaton for player i is a tuple $\mathcal{A}_i = (Q, \delta, o, I)$ where*

- *Q is the set of states,*
- *$\delta : (Q \times W \times \Sigma) \to 2^Q$ is a nondeterministic transition function,*
- *$o : (Q \times W^i) \to \Sigma$ is the output or advice function,*
- *$I \subseteq Q$ is a set of initial states.*

We think of the language accepted by the automaton as a set of strategies of player i. Let μ be a strategy of player i and $T_\mu = (S_\mu, \Rightarrow_\mu, s_0)$ be the corresponding strategy tree. A run of $\mathcal{A}$ on μ is a Q labelled tree $\mathfrak{R} = (S_\mu, \Rightarrow_\mu, s_0, \widehat{\lambda}_\mu, l)$, where l maps each tree node to a state in Q as follows: $l(s_0) \in I$, and for any $s \in S_\mu$ where $s \overset{a}{\Rightarrow}_\mu s'$, we have $l(s') \in \delta(l(s), last(s), a)$.

A run tree $\mathfrak{R}$ is accepted by $\mathcal{A}_i$ if for every tree node $s \in S_\mu$ such that $s \in S^i$ we have $s \overset{a}{\Rightarrow} s'$ implies $o(l(s), last(s)) = a$. A strategy μ is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on μ. For a state q and a tree node s, we often use the notation $o(q, s)$ to denote $o(q, last(s))$.

3.1.4 Strategy specifications to partial strategies

From the semantics, it is easy to see that each strategy specification defines a set of strategies. We now show that it is a *regular* set, recognizable by a finite state

device. More precisely, we show that given any strategy specification σ of player i we can construct a nondeterministic advice automata $\mathcal{A}_\sigma$ such that the language of $\mathcal{A}_\sigma$ is the set of all strategies conforming to the specification σ.

For the rest of the thesis we use the term advice automata to refer to nondeterministic advice automata, if we are considering deterministic advice automata then this fact will be specifically mentioned.

Definition 3.1.4 *For a formula $\psi \in Past(P)$ we define the sub-formula closure of ψ, denoted $CL(\psi)$. Let $CL'(\psi)$ be the least set of formulas such that:*

- $\psi \in CL'(\psi)$.
- *If* $\neg\psi_1 \in CL'(\psi)$ *then* $\psi_1 \in CL'(\psi)$.
- *If* $\psi_1 \vee \psi_2 \in CL'(\psi)$ *then* $\psi_1, \psi_2 \in CL'(\psi)$.
- *If* $\diamondminus\psi_1 \in CL'(\psi)$ *then* $\psi_1 \in CL'(\psi)$.

$CL(\psi) = CL'(\psi) \cup \{\neg\psi_1 \mid \psi_1 \in CL'(\psi)\}$ *where we identify* $\neg\neg\psi_1$ *with* ψ_1.

Definition 3.1.5 *A set* $\mathfrak{t} \subseteq CL(\psi)$ *is called an* atom *if*

- *for all* $\neg\psi_1 \in CL(\psi)$, $\neg\psi_1 \in \mathfrak{t}$ *iff* $\psi_1 \notin \mathfrak{t}$.
- *for all* $\psi_1 \vee \psi_2 \in CL(\psi)$, $\psi_1 \vee \psi_2 \in \mathfrak{t}$ *iff* $\psi_1, \psi_2 \in \mathfrak{t}$.
- *for all* $\diamondminus\psi_1 \in CL(\psi)$, *if* $\psi_1 \in \mathfrak{t}$ *then* $\diamondminus\psi_1 \in \mathfrak{t}$.

In other words, a subset of formulas in the closure is called an atom if it is propositionally consistent and complete. For a formula ψ, let $AT(\psi)$ denote the set of all atoms of ψ and let $\mathcal{C}_0 = \{\mathfrak{t} \in AT(\psi) \mid$ for all $\diamondminus\psi_1 \in CL(\psi)$ if $\diamondminus\psi_1 \in \mathfrak{t}$ then $\psi_1 \in \mathfrak{t}\}$. We define a transition relation on $AT(\psi)$ as: $\mathfrak{t} \rightarrow_{AT} \mathfrak{t}'$ iff for all $\diamondminus\psi_1 \in CL(\psi)$ the following conditions hold.

- if $\diamondminus\psi_1 \in \mathfrak{t}$ then $\diamondminus\psi_1 \in \mathfrak{t}'$.
- if $\diamondminus\psi_1 \in \mathfrak{t}'$ and $\psi_1 \notin \mathfrak{t}'$ then $\diamondminus\psi_1 \in \mathfrak{t}$.

We translate a strategy specification to an advice automaton in an inductive manner. For each atomic specification we first construct an advice automaton and then compose these automata. The following lemma illustrates the construction for atomic strategy specifications.

Lemma 3.1.6 *Given an atomic strategy specification $\sigma = [\psi \mapsto a]^i$ of player i we can construct an advice automaton $\mathcal{A}_\sigma$ such that for a game arena $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ and $\mu \in \Omega^i(\mathcal{G})$ we have $\mu \in Lang(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.*

Proof: Intuitively, the constructed automaton works as follows. Its states are atoms of ψ which keep track of the past formulas satisfied along a play and ensure that the valuations are consistent. At a player i game position, if ψ is in the current atom then the automaton specifies the output a as advice. We give the formal definition below. Let $\sharp$ be a symbol not occurring in Σ. The advice automaton $\mathcal{A}_\sigma = (Q_\sigma, \delta_\sigma, o_\sigma, I_\sigma)$, where

- $Q_\sigma = AT(\psi) \times \Sigma$.
- $Q_\sigma = \{(\mathfrak{t}, x) \mid \mathfrak{t} \in \mathcal{C}_0, x \in \Sigma\}$.
- $\delta_\sigma : (Q_\sigma \times W \times \Sigma) \rightarrow 2^{Q_\sigma}$ is defined as follows: for all $(\mathfrak{t}, x) \in Q_\sigma$, for all $w \in W$ and $b \in \Sigma$, we have $(\mathfrak{t}', y) \in \delta_\sigma((\mathfrak{t}, x), w, b)$ iff
  - $V(w) = \mathfrak{t} \cap Voc(\psi)$.
  - $\mathfrak{t} \rightarrow_{AT} \mathfrak{t}'$.
- $o((\mathfrak{t}, x), w) = \begin{cases} a & \text{if } V(w) = \mathfrak{t} \cap Voc(\psi) \text{ and } \psi \in \mathfrak{t} \\ x & \text{if } V(w) = \mathfrak{t} \cap Voc(\psi) \text{ and } \psi \notin \mathfrak{t} \\ \sharp & \text{if } V(w) \neq \mathfrak{t} \cap Voc(\psi) \end{cases}$

For the automaton $\mathcal{A}_\sigma$ constructed above, we show that $\mu \in Lang(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.
($\Rightarrow$) Suppose $\mu = (S_\mu, \Rightarrow_\mu, s_0, \widehat{\lambda}_\mu) \in Lang(\mathcal{A}_\sigma)$. Let $\mathfrak{R} = (S_\mu, \Rightarrow_\mu, s_0, \widehat{\lambda}_\mu, l)$ be a corresponding Q-labelled tree accepted by $\mathcal{A}_\sigma$. We need to show that for all $s \in S_\mu$ such that $\widehat{\lambda}(s) = i$, we have $\rho^s_{s_0}, s \models \psi$ implies $out_{\rho^s_{s_0}}(s) = a$. Note that since T is an accepting run, we have for all s, $o(l(s), last(s)) \neq \sharp$.

The following claim, asserts that the states of the automaton check the past requirements correctly. Below we use the notation $\psi \in (\mathfrak{t}, x)$ to mean $\psi \in \mathfrak{t}$.
Claim : For all $s \in S_\mu$, for all $\psi' \in CL(\psi)$, $\psi' \in l(s)$ iff $\rho_s, s \models \psi'$.

The claim can be shown by induction on the structure of ψ.
($\psi = p \in P$) : We have $p \in l(s)$ iff $p \in V(s)$ (since $o(l(s), last(s)) \neq \sharp$) iff $\rho_s, s \models p$. Recall that $V(s) = V(last(s))$.
When ψ is of the form $\neg\psi_1$ and $\psi_1 \vee \psi_2$, the claim easily follows by applying induction hypothesis.

($\psi = \Diamond\psi_1$): Let $\rho_s = s_0 a_0 \dots s_{k-1} a_{k-1} s_k = s$ and the Q labelling on ρ_s be the sequence $(\mathfrak{t}_0, x_0)\dots(\mathfrak{t}_{k-1}, x_{k-1})(\mathfrak{t}_k, x_k) = (\mathfrak{t}, x)$. Suppose $\Diamond\psi_1 \in l(s)$. If $\psi_1 \in \mathfrak{t}$ then the claim follows by applying induction hypothesis. Otherwise, by the definition of the transition relation we have $\Diamond\psi_1 \in \mathfrak{t}_{k-1}$. Again if $\psi_1 \in \mathfrak{t}_{k-1}$ then we are done. By definition we have for all $\Diamond\psi' \in \mathfrak{t}_0$, $\psi' \in \mathfrak{t}_0$. Therefore by repeating the above argument, we get that there exists $j : 0 \leq j \leq k$ such that $\psi_1 \in \mathfrak{t}_j$. By induction hypothesis, $\rho_s, s_j \models \psi_1$. By semantics we have $\rho_s, s \models \Diamond\psi_1$.

To see the converse, suppose $\rho_s, s \models \Diamond\psi_1$. By semantics, there exist $j : 0 \leq j \leq k$ such that $\rho_s, s_j \models \psi_1$. By induction hypothesis, $\psi_1 \in \mathfrak{t}_j$ and by definition of atom $\Diamond\psi_1 \in \mathfrak{t}_j$. If $j = k$ we are done. Otherwise by definition of the transition relation we get $\Diamond\psi_1 \in \mathfrak{t}_{j+1}$. By repeating the argument we get $\Diamond\psi_1 \in \mathfrak{t}$.

End of claim

From the above claim, we have $\rho_s, s \models \psi$ implies $\psi \in l(s)$. By the definition of the output function o, we have $o(l(s), s) = a$.
($\Leftarrow$) Suppose $\mu \models_i [\psi \mapsto a]^i$. From the semantics, we have $\forall s \in S_\mu$ such that $\widehat{\lambda}(s) = i$ if $\rho^s_{s_0}, s \models \psi$ then $out_{\rho^s_{s_0}}(s) = a$. We need to show that there exists a Q-labelled tree accepted by $\mathcal{A}_\sigma$. Consider the labelling function defined as follows. Fix any $x_0 \in \Sigma$.

- For $s \in S^i_\mu$, let $l(s) = (\{\psi' \in CL(\psi) \mid \rho_s, s \models \psi'\}, out_{\rho^s_{s_0}}(s))$.

- For $s \in S^{\overline{i}}_\mu$, let $l(s) = (\{\psi' \in CL(\psi) \mid \rho_s, s \models \psi'\}, x_0)$.

It is easy to check that for all $s \in S_\mu$, $l(s) = (C, x)$ where C constitutes an atom and the transition relation is respected. What remains to be shown is the following:

- for all $s \in S_\mu$ such that $\widehat{\lambda}(s) = i$ we have $o(l(s), last(s)) = out_{\rho^s_{s_0}}(s)$.

Consider any $s \in S_\mu$ with $\widehat{\lambda}(s) = i$. If $\rho^s_{s_0} \models \psi$ then by the above claim $\psi \in l(s)$. By definition of the output function $o(l(s, last(s)) = a$. If $\rho^s_{s_0} \not\models \psi$ then by definition of the labelling function we have $l(s) = (\mathfrak{t}, y)$ where $y = out_{\rho^s_{s_0}}(s)$. Thus by definition of the output function of the automaton we get $o(l(s, last(s)) = y = out_{\rho^s_{s_0}}(s)$.
□

Lemma 3.1.7 *Given a strategy specification $\sigma \in Strat^i(P^i)$ of player i, we can construct an advice automaton $\mathcal{A}_\sigma$ such that for a game arena $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ and $\mu \in \Omega^i(\mathcal{G})$ we have $\mu \in Lang(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.*

Proof: We proceed by induction on the structure of σ. The base case when $\sigma = [\psi \mapsto a]^i$ follows from Lemma 3.1.6.
($\sigma = \sigma_1 \cdot \sigma_2$): By induction hypothesis there exist $\mathcal{A}_{\sigma_1} = (Q_{\sigma_1}, \delta_{\sigma_1}, o_{\sigma_1}, I_{\sigma_1})$ and $\mathcal{A}_{\sigma_2} = (Q_{\sigma_2}, \delta_{\sigma_2}, o_{\sigma_2}, I_{\sigma_2})$ which accept all strategies satisfying σ_1 and σ_2 respectively. To obtain an automaton which accepts all strategies which satisfy $\sigma_1 \cdot \sigma_2$ we just need to take the product of $\mathcal{A}_{\sigma_1}$ and $\mathcal{A}_{\sigma_2}$.
($\sigma = \sigma_1 + \sigma_2$): Inductively we have automata $\mathcal{A}_{\sigma_1}$ and $\mathcal{A}_{\sigma_2}$. The advice automaton for $\mathcal{A}_\sigma$ simulates both $\mathcal{A}_{\sigma_1}$ and $\mathcal{A}_{\sigma_2}$ in parallel and at any player i game position, nondeterministically chooses to output the advice of either $\mathcal{A}_{\sigma_1}$ or $\mathcal{A}_{\sigma_2}$.
($\sigma = \pi \Rightarrow \sigma'$): By induction hypothesis we have $\mathcal{A}_\pi = (Q_\pi, \delta_\pi, o_\pi, I_\pi)$ which accepts all player $\overline{i}$ strategies satisfying π and $\mathcal{A}_{\sigma'} = (Q_{\sigma'}, \delta_{\sigma'}, o_{\sigma'}, I_{\sigma'})$ which accepts all player i strategies satisfying σ'.

The automaton $\mathcal{A}_\sigma$ has the product states of $\mathcal{A}_\pi$ and $\mathcal{A}_{\sigma'}$ as its states along with a special state q_{free}. The automaton keeps simulating both $\mathcal{A}_\pi$, $\mathcal{A}_{\sigma'}$ and keeps checking if the path violates the advice given by $\mathcal{A}_\pi$, if so it moves into state q_{free} from which point onwards it is "free" to produce any advice. Unless π is violated, it is forced to follow the transitions of $\mathcal{A}_{\sigma'}$.

Define $\mathcal{A}_\sigma = (Q, \delta, o, I)$ where $Q = (Q_\pi \times Q_{\sigma'}) \cup (\{q_{free}\} \times \Sigma)$. The transition function is given as follows:

- For $s \in S^i_\mu$, we have $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_1, q_2) \mid q_1 \in \delta_\pi(q_\pi, s, a)$ and $q_2 \in \delta_{\sigma'}(q_{\sigma'}, s, a)\}$.
- For $s \in S^{\overline{i}}_\mu$, we have:
  - If $o_\pi(q_\pi, s) \neq a$, then $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_{free}, a) \mid a \in \Sigma\}$.
  - If $o_\pi(q_\pi, s) = a$, then $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_1, q_2) \mid q_1 \in \delta_\pi(q_\pi, s, a)$ and $q_2 \in \delta_{\sigma'}(q_{\sigma'}, s, a)\}$.
- $\delta((q_{free}, x), s, a) = \{(q_{free}, a) | a \in \Sigma\}$

For $s \in S^i_\mu$, the output function is defined as: $o((q_\pi, q_{\sigma'}), last(s)) = o_{\sigma'}(q_{\sigma'}, last(s))$ and $o((q_{free}, x), last(s)) = x$. □

The following proposition can then be easily shown using Lemma 3.1.7.

Proposition 3.1.8 *Given a game arena $\mathcal{G}$, player $i \in N$ and a strategy specification $\sigma \in Strat^i(P^i)$, we can construct an advice automaton $\mathcal{A}_\sigma$ such that $Lang(\mathcal{A}_\sigma) \cap \Omega^i(\mathcal{G}) = \mathcal{TL}(\mathcal{G}, \sigma)$.*

3.2 Remarks on strategy specifications

A relatively simple syntax for strategy specifications was introduced in Section 3.1.1. The objective was to illustrate the concept and it should be noted that the above mentioned results can be extended to any set of specifications that allows an effective automaton construction. However, operators are best added after a systematic study of their algebraic properties. An extension of particular interest is that of multi-stage games.

Multi-stage games: So far we have viewed the game arena as representing a single unbounded duration game. We can also reason about multi-stage games using similar techniques. To model such games, we fix a finite set of colours *Col* and associate the game positions with elements of this set. The idea being that the change of colour indicates the switch from one stage to the next. For simplicity, we also assume that each stage has a unique *start* and *end* game position along with a unique action *switch* such that $start \xrightarrow{switch} start$. The set of strategy specifications for player i can be extended using the following constructs.

- $\mathcal{Y}\pi \Rightarrow \sigma$ - if in the previous stage player $\overline{i}$ conforms to π then play according to σ.
- $\mathcal{P}\pi \Rightarrow \sigma$ - if there is a previous stage where the player $\overline{i}$ played according to π then play σ.
- $\mathcal{H}\pi \Rightarrow \sigma$ - if player $\overline{i}$ in all the previous stages has conformed to π then play according to σ

It is quite straight forward to give the formal semantics of the above constructs and therefore we do not take it up here. It is also relatively easy to verify that the constructs can be compiled into a finite state automaton. Thus an equivalent of lemma 3.1.7 can be shown for the extended syntax as well. Below we show that this extended set of strategy specifications can be effectively used to reason about multi-stage games.

Iterated prisoner's dilemma: Prisoner's dilemma [OR94] is the classic example portraying the weakness of equilibrium notions. The unique equilibrium in this game is where both prisoners defect, when cooperation would have resulted in a better

payoff for both. This game illustrates the fact that equilibrium solutions need not always give the most efficient outcomes for players.

Axelrod was interested in finding out more about the strategies which were successful in playing prisoner's dilemma *in practice* [Axe84]. He conducted a tournament where strategies were made to compete against each other. Note that this is a departure from the standard analysis in terms of solution concepts like equilibrium, where the emphasis remains on trying to figure out what happens if a player unilaterally deviates. It turned out that the winner was a simple strategy called "tit-for-tat" which cooperates in the first round and in subsequent rounds mimics what the opponent did in the previous round.

The iterated version of prisoner's dilemma can be easily modelled as a multistage game. Let c, d correspond to the actions "cooperate", "defect" respectively and let *init* be a proposition which holds only at the first stage. We also use the abbreviation $play(i, a) = [True \mapsto a]^i$ for $i \in N$ and $a \in \Sigma$. The strategy tit-for-tat can be expressed in the extended strategy specification syntax as follows.

- $\sigma_{\mathbf{TFT}} = (\mathcal{Y}\ play(2, d) \Rightarrow [\neg init \mapsto d]^1) \cdot (\mathcal{Y}\ play(2, c) \Rightarrow [\neg init \mapsto c]^1)$.
- Tit-for-tat: $[init \mapsto c]^1 \cdot \sigma_{\mathbf{TFT}}$.

The strategy "grim" (also called "trigger") which cooperates till the opponent defects and then defects forever can be expressed as

- Grim: $(\mathcal{H}\ play(2, c) \Rightarrow play(1, c)) \cdot (\mathcal{P}\ play(2, d) \Rightarrow [\neg init \mapsto d]^1)$.

Note that tit-for-tat and grudge are examples of complete strategies. On further analysis, it was noticed that the top scoring strategies satisfied certain properties in common. It turns out that the exact strategy is not particularly important and any strategy satisfying these properties would have performed well in the tournament. The properties identified were as follows:

- Niceness: Is not the first to defect.
- Forgiveness: Does not hold a grudge once the opponent cooperates.
- Retaliatory: If the opponent defects, punishes him by defecting.

The idea behind strategy specification exactly corresponds to expressing such properties of strategies rather than complete strategies. For prisoner's dilemma, this can be achieved as follows:

- Niceness: $[init \mapsto c]^1 \cdot (\mathcal{H}\ play(2,c) \Rightarrow [\neg init \mapsto c]^1)$.
- Forgiveness: $[init \mapsto c]^1 \cdot (\mathcal{Y}\ play(2,c) \Rightarrow [\neg init \mapsto c]^1)$.
- Retaliatory: $[init \mapsto c]^1 \cdot (\mathcal{Y}\ play(2,d) \Rightarrow [\neg init \mapsto d]^1)$.

3.3 A strategy logic

We now discuss how we may embed structured strategies in a formal logic. Formulas of the logic (also referred to as *game formulas*) are built up using structured strategy specifications (as defined in section 3.1). Game formulas describe the game arena in a standard modal logic, and in addition specify the result of a player following a particular strategy at a game position, to choose a specific move a. Using these formulas one can specify how a strategy helps to eventually *win* (ensure) an outcome β.

3.3.1 Syntax

The syntax of the logic is given by:

$$\Pi := p \in P \mid (\sigma)_i : c \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle a \rangle \alpha \mid \langle \overline{a} \rangle \alpha \mid \Diamond\!\!\!-\alpha \mid \sigma \rightsquigarrow_i \beta$$

where $a, c \in \Sigma$, $\sigma \in Strat^i(P^i)$, $\beta \in Past(P^i)$. The derived connectives $\wedge$, $\supset$ and $[a]\alpha$ are defined as usual. Let $\boxminus\alpha = \neg\Diamond\!\!\!-\neg\alpha$, $\langle\mathcal{N}\rangle\alpha = \bigvee_{a\in\Sigma} \langle a\rangle\alpha$, $[\mathcal{N}]\alpha = \neg\langle\mathcal{N}\rangle\neg\alpha$, $\langle\mathcal{P}\rangle\alpha = \bigvee\limits_{a\in\Sigma} \langle\overline{a}\rangle\alpha$ and $[\mathcal{P}] = \neg\langle\mathcal{P}\rangle\neg\alpha$.

The formula $\langle a\rangle\alpha$ talks about one step in the future. It asserts the existence of an a edge after which α holds. Note that future time assertions up to any bounded depth can be coded by iteration of this construct. $\langle\overline{a}\rangle\alpha$ is the corresponding construct to refer to one step in the past. The formula $\Diamond\!\!\!-\alpha$ makes assertion about the unbounded past, it specifies the transitive closure of the one step past operator. Since a strategy specification can base its advice on apparent behaviour of players in the past, the past time modalities turn out to be useful in logical reasoning.

The formula $(\sigma)_i : c$ asserts, at any game position, that the strategy specification σ for player i suggests that the move c can be played at that position. The formula $\sigma \rightsquigarrow_i \beta$ says that from this position, following the strategy σ for player i ensures the outcome β. These two modalities constitute the main constructs of our logic.

3.3.2 Semantics

Model: Models of the logic consist of extensive form game trees along with a valuation function. A model $M = (T, V)$ where $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ is an extensive form game tree and $V : S \rightarrow 2^P$ is a valuation function. As mentioned earlier, we require that the valuation function satisfies the condition:

- For all $s \in S$ and $i \in N$, $\mathbf{turn}_i \in V(s)$ iff $\widehat{\lambda}(s) = i$.

For a node $s \in S$, let $moves(s) = \{a \in \Sigma \mid \exists s' \in S \text{ with } s \stackrel{a}{\Rightarrow} s'\}$. For the purpose of defining the logic it is convenient to define the notion of the set of moves enabled by a strategy specification σ at a game position s (denoted $\sigma(s)$).

Definition 3.3.1 *For a game tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ and a game position s, let $\rho^s_{s_0} : s_0 \stackrel{a_1}{\Rightarrow} s_1 \cdots \stackrel{a_m}{\Rightarrow} s_m = s$ denote the unique path from s_0 to s. For a strategy specification $\sigma \in Strat^i(P^i)$ we define $\sigma(s)$ as follows:*

- $[\psi \mapsto a]^i(s) = \begin{cases} \{a\} & \text{if } \widehat{\lambda}(s) = i \text{ and } \rho^s_{s_0}, m \models \psi. \\ \Sigma & \text{otherwise.} \end{cases}$
- $(\sigma_1 + \sigma_2)(s) = \sigma_1(s) \cup \sigma_2(s)$.
- $(\sigma_1 \cdot \sigma_2)(s) = \sigma_1(s) \cap \sigma_2(s)$.
- $(\pi \Rightarrow \sigma)(s) = \begin{cases} \sigma(s) & \text{if } \forall j : 0 \leq j < m,\ a_j \in \pi(s_j). \\ \Sigma & \text{otherwise.} \end{cases}$

We say that a path $\rho^{s'}_s : s = s_1 \stackrel{a_1}{\Rightarrow} s_2 \cdots \stackrel{a_{m-1}}{\Rightarrow} s_m = s'$ in T conforms to σ if $\forall j : 1 \leq j < m$, $a_j \in \sigma(s_j)$. When the path constitutes a proper play, i.e. when $s = s_0$, we say that the play conforms to σ. The following proposition is easy to see from the definition.

Proposition 3.3.2 *Given a strategy $\mu = (S_\mu, \Rightarrow_\mu, s_0, \widehat{\lambda}_\mu)$ for player i along with a specification σ, $\mu \models_i \sigma$ (as defined in section 3.1) iff for all $s \in S_\mu$ such that $\widehat{\lambda}_\mu(s) = i$ we have $out_{\rho^s_{s_0}}(s) \in \sigma(s)$.*

For a game tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ and a node $s \in S$, let T_s denote the tree which consists of the unique path $\rho^s_{s_0}$ and the subtree rooted at s. For a strategy specification $\sigma \in Strat^i(P^i)$, we define $T_s \restriction \sigma = (S_\sigma, \Rightarrow_\sigma, s_0, \widehat{\lambda}_\sigma)$ to be the least subtree of T_s which contains the unique path from s_0 to s and satisfies the property: for every $s_1 \in S_\sigma$,

- if $\widehat{\lambda}_\sigma(s_1) = i$ then for all s_2 with $s_1 \stackrel{a}{\Rightarrow} s_2$ and $a \in \sigma(s_1)$ we have $s_1 \stackrel{a}{\Rightarrow}_\sigma s_2$ and $\widehat{\lambda}_\sigma(s_2) = \widehat{\lambda}(s_2)$.
- if $\widehat{\lambda}_\sigma(s_1) = \overline{i}$ then for all s_2 with $s_1 \stackrel{a}{\Rightarrow} s_2$ we have $s_1 \stackrel{a}{\Rightarrow}_\sigma s_2$ and $\widehat{\lambda}_\sigma(s_2) = \widehat{\lambda}(s_2)$.

The truth of a formula $\alpha \in \Pi$ in a model M and position s (denoted $M, s \models \alpha$) is defined by induction on the structure of α, as usual. Let $\rho^s_{s_0}$ be $s_0 \stackrel{a_0}{\Rightarrow} s_1 \cdots \stackrel{a_{m-1}}{\Rightarrow} s_m = s$.

- $M, s \models p$ iff $p \in V(s)$.
- $M, s \models \neg\alpha$ iff $M, s \not\models \alpha$.
- $M, s \models \alpha_1 \vee \alpha_2$ iff $M, s \models \alpha_1$ or $M, s \models \alpha_2$.
- $M, s \models \langle a \rangle \alpha$ iff there exists s' such that $s \stackrel{a}{\Rightarrow} s'$ and $M, s' \models \alpha$.
- $M, s \models \langle \overline{a} \rangle \alpha$ iff $m > 0$, $a = a_{m-1}$ and $M, s_{m-1} \models \alpha$.
- $M, s \models \Diamond\!\!\!\!-\alpha$ iff there exists $j : 0 \leq j \leq m$ such that $M, s_j \models \alpha$.
- $M, s \models (\sigma)_i : c$ iff $c \in \sigma(s)$.
- $M, s \models \sigma \rightsquigarrow_i \beta$ iff for all s' such that $s \Rightarrow^*_\sigma s'$ in $T_s \upharpoonright \sigma$, we have $M, s' \models \beta \wedge (\mathbf{turn}_i \supset \mathit{enabled}_\sigma)$.

where $\mathit{enabled}_\sigma = \bigvee_{a \in \Sigma} (\langle a \rangle \mathit{True} \wedge (\sigma)_i : a)$.

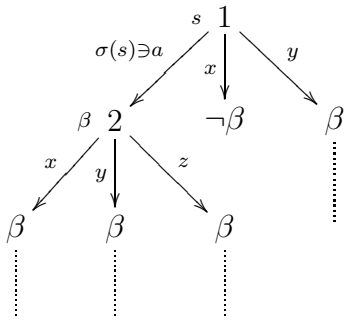


Figure 3.3: Interpretation of $\sigma \rightsquigarrow_i \beta$

Figure 3.3 illustrates the semantics of $\sigma \rightsquigarrow_1 \beta$. It says, for any 1 node β is ensured by playing according to σ; for a 2 node, all actions should ensure β.

The notions of satisfiablility and validity can be defined in the standard way. A formula α is satisfiable iff there exists a model M and s such that $M, s \models \alpha$.

A formula α is said to be valid iff for all models M and for all nodes s, we have $M, s \models \alpha$.

Consider the formula $null^i \rightsquigarrow_i \beta$. This asserts that player i has a strategy to ensure β no matter what player $\overline{i}$ does. This makes no reference to how player i may achieve this objective and thus, is similar to assertions in most game logics. Now consider the formula $\sigma \rightsquigarrow_i \beta$. This says something stronger: that there exists a strategy μ satisfying σ for player i such that irrespective of what player $\overline{i}$ plays, β is guaranteed. Here, the mechanism μ used by player i to ensure β is specified by the property σ.

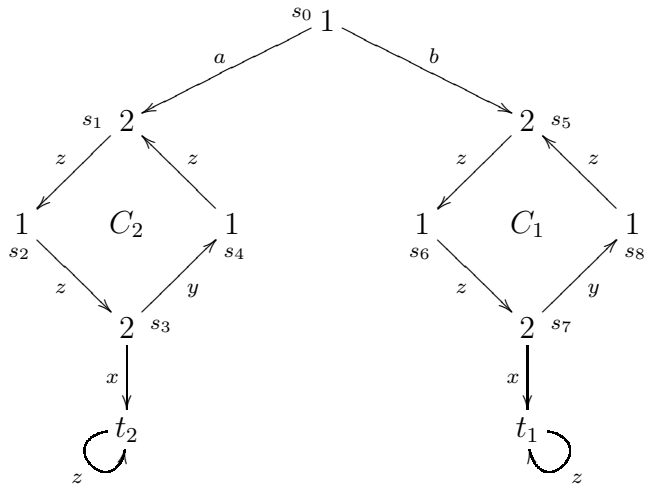


Figure 3.4: Game arena

Example 3.3.3 Consider the game obtained by the unfolding of the arena shown in Figure 3.4. Players alternate moves with 1 starting at s_0. There are two cycles $C_1 : s_5 \to s_6 \to s_7 \to s_8 \to s_5$, $C_2 : s_1 \to s_2 \to s_3 \to s_4 \to s_1$ and self loops on nodes t_1 and t_2. Let the preference ordering of player 1 be $t_1 \preceq^1 t_2 \preceq^1 C_2 \preceq^1 C_1$. As far as player 2 is concerned $t_1 \preceq^2 C_1$ and he is indifferent between C_2 and t_2. However, he prefers C_2 or t_2 over $\{C_1, t_1\}$. Equilibrium reasoning will advise player 1 to choose the action "b" at s_0 since at position s_7 it is irrational for player 2 to move x as it results in his worst outcome. However the utility difference between C_1 and t_1 for player 2 might be negligible compared to the incentive of staying in the "left" path. Therefore player 2 might decide to punish player 1 for moving b when player 1 knew that $\{C_2, t_2\}$ was equally preferred by player 2. Even though t_1 is

the worst outcome, at s_7 player 2 can play x to implement the punishment. Let the valuation satisfy the following constraints:

- The proposition p_j holds at states $\{s_3, s_7\}$,
- p_{init} holds at state s_0,
- p_{good} holds at states $\{s_0, s_1, s_2, s_3, s_4\}$
- p_{punish} holds at states $\{s_0, s_5, s_6, s_7, t_1\}$.

The local objective of player 2 is to remain on the good path or to implement the punishment. Player 2's strategy specification can be written as

$$\pi = ([p_{init} \mapsto b]^1 \Rightarrow [p_j \mapsto x]^2) \cdot ([p_{init} \mapsto a]^1 \Rightarrow [p_j \mapsto y]^2).$$

We get that $\pi \rightsquigarrow_2 (p_{good} \vee p_{punish})$. If player 1 knows player 2's strategy, she might be tempted to play "a" at s_0 by which the play ends up in C_2. Let the proposition p_{worst} hold at t_1 which is the worst outcome for player 1. Then we have $[p_{init} \mapsto a]^1 \rightsquigarrow_1 \neg p_{worst}$. This says that if player 1 chooses a at the initial position then she can ensure that the worst outcome is avoided. □

Example 3.3.4 Consider the formula $null^i \rightsquigarrow_i (\pi \rightsquigarrow_{\overline{i}} \beta)$. This says that player i has a strategy such that from all the outcome states if player $\overline{i}$ follows a strategy corresponding to π then β can be ensured. In other words this construct can be used to make the assertion: given that player $\overline{i}$ sticks to the specification π, there is a strategy for player i to make sure that the maximum $\overline{i}$ can achieve is β. In general, the assertion can be parameterized by a specification of player i as well. This takes the form $\sigma \rightsquigarrow_i (\pi \rightsquigarrow_{\overline{i}} \beta)$. □

3.3.3 Axiom system

We now present our axiomatization of the valid formulas of the logic. We find the following abbreviations useful:

- $root = \neg\langle\mathcal{P}\rangle\, True$ defines the root node to be one that has no predecessors.
- $\delta_i^\sigma(a) = \mathbf{turn}_i \wedge (\sigma)_i : a$ denotes that move "a" is enabled by σ at an i node.
- $inv_i^\sigma(a, \beta) = (\mathbf{turn}_i \wedge (\sigma)_i : a) \supset [a](\sigma \rightsquigarrow_i \beta)$ denotes the fact that after an "a" move by player i which conforms to σ, $\sigma \rightsquigarrow_i \beta$ continues to hold.

- $inv_{\overline{i}}^{\sigma}(\beta) = \mathbf{turn}_{\overline{i}} \supset [\mathcal{N}](\sigma \rightsquigarrow_i \beta)$ says that after any move of $\overline{i}$, $\sigma \rightsquigarrow_i \beta$ continues to hold.
- $conf_{\pi} = \boxminus(\langle\overline{a}\rangle\mathbf{turn}_{\overline{i}} \supset \langle\overline{a}\rangle(\pi)_{\overline{i}} : a)$ denotes that all opponent moves in the past conform to π.

The axiom schemes

(A0) All the substitutional instances of the tautologies of propositional calculus.

(A1) (a) $[a](\alpha_1 \supset \alpha_2) \supset ([a]\alpha_1 \supset [a]\alpha_2)$

(b) $[\overline{a}](\alpha_1 \supset \alpha_2) \supset ([\overline{a}]\alpha_1 \supset [\overline{a}]\alpha_2)$

(A2) (a) $\langle a\rangle\alpha \supset [a]\alpha$

(b) $\langle\overline{a}\rangle\alpha \supset [\overline{a}]\alpha$

(c) $\langle\overline{a}\rangle \mathit{True} \supset \neg\langle\overline{b}\rangle \mathit{True}$ for all $b \neq a$

(A3) (a) $\alpha \supset [a]\langle\overline{a}\rangle\alpha$

(b) $\alpha \supset [\overline{a}]\langle a\rangle\alpha$

(A4) (a) $\diamondminus \mathit{root}$

(b) $\boxminus\alpha \equiv (\alpha \wedge [\mathcal{P}]\boxminus\alpha)$

(A5) (a) $([\psi \mapsto a]^i)_i : a$ for all $a \in \Sigma$

(b) $\mathbf{turn}_i \wedge ([\psi \mapsto a]^i)_i : c \equiv \neg\psi$ for all $a \neq c$

(A6) (a) $(\sigma_1 + \sigma_2)_i : c \equiv \sigma_1 : c \vee \sigma_2 : c$

(b) $(\sigma_1 \cdot \sigma_2)_i : c \equiv \sigma_1 : c \wedge \sigma_2 : c$

(c) $(\pi \Rightarrow \sigma)_i : c \equiv \mathit{conf}_{\pi} \supset (\sigma)_i : c$

(A7) $\sigma \rightsquigarrow_i \beta \supset (\beta \wedge \mathit{inv}_i^{\sigma}(a, \beta) \wedge \mathit{inv}_{\overline{i}}^{\sigma}(\beta) \wedge \mathit{enabled}_{\sigma})$

Inference rules

$$(MP)\ \frac{\alpha,\quad \alpha \supset \beta}{\beta} \qquad (NG)\ \frac{\alpha}{[a]\alpha} \qquad (NG\text{-})\ \frac{\alpha}{[\overline{a}]\alpha}$$

$$(\mathit{Ind\text{-}past})\ \frac{\alpha \supset [P]\alpha}{\alpha \supset \boxminus\alpha}$$

$$(\mathit{Ind} \rightsquigarrow)\ \frac{\alpha \wedge \delta_i^{\sigma}(a) \supset [a]\alpha,\quad \alpha \wedge \mathbf{turn}_{\overline{i}} \supset [\mathcal{N}]\alpha,\quad \alpha \supset \beta \wedge \mathit{enabled}_{\sigma}}{\alpha \supset \sigma \rightsquigarrow_i \beta}$$

The axioms are mostly standard. After the Kripke axioms for the $\langle a \rangle$ modalities, we have axioms that ensure determinacy of both $\langle a \rangle$ and $\langle \overline{a} \rangle$ modalities, and an axiom to assert the uniqueness of the latter. We then have axioms that relate the previous and next modalities with each other, as well as to assert that the past modality steps through the $\langle \overline{a} \rangle$ modality. An axiom asserts the existence of the root in the past. The rest of the axioms describe the semantics of strategy specifications.

The rule *Ind-past* is standard, while $Ind \rightsquigarrow$ illustrates the new kind of reasoning in the logic. It says that to infer that the formula $\sigma \rightsquigarrow_i \beta$ holds in all reachable states, β must hold at the asserted state and

- for a player i node after every move which conforms to σ, β continues to hold.
- for a player $\overline{i}$ node after every enabled move, β continues to hold.
- player i does not get stuck by playing σ.

3.3.4 Soundness

The validity of axioms (A1) to (A3b) and (A4b) can be shown using standard modal techniques. Axiom (A4a) is valid since the formulas are interpreted over a finite extensive game tree with a unique root. The validity axioms (A5) and (A6) can be easily verified using the semantics of the logic.

Proposition 3.3.5 *Axiom (A7) is valid.*

Proof: Suppose axiom (A7) is not valid. Then there exists a node s such that $M, s \models \sigma \rightsquigarrow_i \beta$ and one of the following holds:

- $M, s \not\models \beta$: In this case, from semantics we get that $M, s \not\models \sigma \rightsquigarrow_i \beta$ which is a contradiction.
- $M, s \not\models inv_i^{\sigma}(a, \beta)$: In this case, we have $s \in W^i$, $M, s \models (\sigma)_i : a$ and $M, s' \not\models \sigma \rightsquigarrow_i \beta$ where $s \overset{a}{\Rightarrow} s'$. This implies that there is a path $\rho_{s'}^{s_k}$ which conforms to σ and either $M, s_k \not\models \beta$ or $moves(s_k) \cap \sigma(s_k) = \emptyset$. But since $s \overset{a}{\Rightarrow} s'$, we have $\rho_s^{s_k}$ conforms to σ as well. From which it follows that $M, s \not\models \sigma \rightsquigarrow_i \beta$ which is a contradiction.
- $M, s \not\models inv_{\overline{i}}^{\sigma}(\beta)$: We have a similar argument as above.
- $M, s \not\models enabled_{\sigma}$: This means $moves(s) \cap \sigma(s) = \emptyset$, by semantics we have $M, s \not\models \sigma \rightsquigarrow_i \beta$ which is a contradiction.

□

Proposition 3.3.6 *Rule Ind* $\rightsquigarrow$ *preserves validity.*

Proof: Suppose that the premise is valid and the conclusion is not. Then for some node s we have $M, s \models \alpha$ and $M, s \not\models \sigma \rightsquigarrow_i \beta$. i.e. there is a path $\rho_s^{s_k}$ which conforms to σ such that $M, s_k \not\models \beta$ or $\sigma(s_k) \cap moves(s_k) = \emptyset$. Let $\rho_s^{s_k}$ be the shortest of such path.

Suppose $M, s_k \not\models \beta$, then we have the following two cases to consider.

- $s_{k-1} \in W^i$: By assumption on the path $\rho_s^{s_k}$, we have $M, s_{k-1} \models \alpha \wedge \delta_i^\sigma(a_{k-1})$. From validity of $\alpha \supset \beta$ (the premise), we have $M, s_k \not\models \alpha$, which implies $M, s_{k-1} \not\models [a_{k-1}]\alpha$. Therefore we get $M, s_{k-1} \not\models (\alpha \wedge \delta_i^\sigma(a_{k-1})) \supset [a_{k-1}]\alpha$, which gives us a contradiction to the validity of a premise.

- $s_{k-1} \in W^{\overline{i}}$: By assumption on the path $\rho_s^{s_k}$, we have $M, s_{k-1} \models \alpha \wedge \mathbf{turn}_{\overline{i}}$. Using an argument similar to the previous case we also get $M, s_{k-1} \not\models [a_{k-1}]\alpha$. Therefore we have $M, s_{k-1} \not\models (\alpha \wedge \mathbf{turn}_{\overline{i}}) \supset [\mathcal{N}]\alpha$, giving us a contradiction to the validity of a premise.

If $\sigma(s_k) \cap moves(s_k) = \emptyset$ then we have $M, s_k \models \alpha$ and $M, s_k \not\models enabled_\sigma$. Therefore $M, s_k \not\models (\alpha \supset enabled_\sigma)$, which is the required contradiction. □

3.3.5 Completeness

To show completeness, we prove that every consistent formula is satisfiable. We use the following definitions.

Definition 3.3.7 *For a strategy specification* $\sigma \in Strat^i(P^i)$ *we define the set of past time formulas occurring in* σ*, denoted* $\mathbb{P}(\sigma)$ *inductively as follows:*

- $\mathbb{P}([\psi \mapsto a]^i) = \{\psi\}$ *for* $a \in \Sigma$.
- $\mathbb{P}(\sigma_1 + \sigma_2) = \mathbb{P}(\sigma_1) \cup \mathbb{P}(\sigma_2)$.
- $\mathbb{P}(\sigma_1 \cdot \sigma_2) = \mathbb{P}(\sigma_1) \cup \mathbb{P}(\sigma_2)$.
- $\mathbb{P}(\pi \Rightarrow \sigma_1) = \mathbb{P}(\pi) \cup \mathbb{P}(\sigma_1)$.

Definition 3.3.8 *For a formula α_0 we define the sub-formula closure of α_0 denoted $CL(\alpha_0)$. Let $CL'(\alpha_0)$ be the least set of formulas such that:*

- α_0, $\Diamond root \in CL'(\alpha_0)$.
- *If* $\neg\alpha \in CL'(\alpha_0)$ *then* $\alpha \in CL'(\alpha_0)$.
- *If* $\alpha_1 \vee \alpha_2 \in CL'(\alpha_0)$ *then* $\alpha_1, \alpha_2 \in CL'(\alpha_0)$.
- *For* $\mathcal{M}$ *of the form* $\langle a\rangle$ *or* $\langle \bar{a}\rangle$ *or* $\Diamond$, *if* $\mathcal{M}\alpha \in CL'(\alpha_0)$ *then* $\alpha \in CL'(\alpha_0)$.
- *If* $(\sigma)_i : a \in CL'(\alpha_0)$ *then* $\mathbb{P}(\sigma) \subseteq CL'(\alpha_0)$.
- *If* $\sigma \rightsquigarrow_i \beta \in CL'(\alpha_0)$ *then* $\beta \in CL'(\alpha_0)$, $inv_i^\sigma(a,\beta) \in CL'(\alpha_0)$, $inv_{\bar{i}}^\sigma(\beta) \in CL'(\alpha_0)$, $enabled_\sigma \in CL'(\alpha_0)$ *and* $\mathbb{P}(\sigma) \subseteq CL'(\alpha_0)$.

$CL(\alpha) = CL'(\alpha_0) \cup \{\neg\alpha \mid \alpha \in CL'(\alpha_0)\}$ *where we identify* $\neg\neg\alpha$ *with* α.

Definition 3.3.9 *A set* $\mathfrak{t} \subseteq CL(\alpha_0)$ *is called an atom if*

- $\Diamond root \in \mathfrak{t}$.
- *for all* $\neg\alpha \in CL(\alpha_0)$, $\neg\alpha \in \mathfrak{t}$ *iff* $\alpha \notin \mathfrak{t}$.
- *for all* $\alpha_1 \vee \alpha_2 \in CL(\alpha_0)$, $\alpha_1 \vee \alpha_2 \in \mathfrak{t}$ *iff* $\alpha_1, \alpha_2 \in \mathfrak{t}$.
- *for all* $\Diamond\alpha \in CL(\alpha_0)$, $\Diamond\alpha \in \mathfrak{t}$ *iff* $\alpha \in \mathfrak{t}$ *or* $\langle\mathcal{P}\rangle\Diamond\alpha \in \mathfrak{t}$.
- *for all* $\sigma \rightsquigarrow_i \beta \in CL(\alpha_0)$ *if* $\sigma \rightsquigarrow_i \beta \in \mathfrak{t}$ *then* $\beta \in \mathfrak{t}$, $inv_i^\sigma(a,\beta) \in \mathfrak{t}$, $inv_{\bar{i}}^\sigma(\beta) \in \mathfrak{t}$ *and* $enabled_\sigma \in \mathfrak{t}$.
- *for all* $\neg(\sigma \rightsquigarrow_i \beta) \in CL(\alpha_0)$ *if* $\neg(\sigma \rightsquigarrow_i \beta) \in \mathfrak{t}$ *then* $\neg enabled_\sigma \in \mathfrak{t}$ *or* $\neg\beta \in \mathfrak{t}$ *or* $(\langle\mathcal{N}\rangle\neg(\sigma \rightsquigarrow_i \beta)) \in \mathfrak{t}$.

Let α_0 be a consistent formula, and let $AT(\alpha_0)$ be the set of atoms of α_0. We let $\mathfrak{t}, \mathfrak{t}'$ range over $AT(\alpha_0)$. Each $\mathfrak{t} \in AT(\alpha_0)$ is a finite set of formulas, we denote the conjunction of all formulas in $\mathfrak{t}$ by $\widehat{\mathfrak{t}}$. For a nonempty subset $X \subseteq AT(\alpha_0)$, we denote by $\widetilde{X}$ the disjunction of all $\widehat{\mathfrak{t}}, \mathfrak{t} \in X$. Define a transition relation on $AT(\alpha_0)$ as follows: $\mathfrak{t} \xrightarrow{a}_{AT} \mathfrak{t}'$ iff $\widehat{\mathfrak{t}} \wedge \langle a\rangle\widehat{\mathfrak{t}'}$ is consistent. Call an atom $\mathfrak{t}$ a *root atom* if there does not exist any atom $\mathfrak{t}'$ such that $\mathfrak{t}' \xrightarrow{a}_{AT} \mathfrak{t}$ for some a.

The following lemmas can be shown using standard modal logic techniques.

Lemma 3.3.10 *For all* $\alpha \in CL(\alpha_0)$ *and* $\mathfrak{t} \in AT(\alpha_0)$ *we have the following:*

1. $\widehat{\mathfrak{t}}$ *is consistent.*

2. *if* $\langle a\rangle\alpha$ *is consistent then* α *is consistent.*

3. *for all* $R \subseteq AT(\alpha_0)$, *and* $\mathfrak{t} \in AT(\alpha_0)$, *if* $\mathfrak{t} \in R$ *then* $\vdash \widehat{\mathfrak{t}} \supset \widetilde{R}$.

4. *if* $\vdash \alpha$ *then* $\alpha \in \mathfrak{t}$.

5. *if* $\vdash \widehat{\mathfrak{t}} \supset \alpha$ *then* $\widehat{\mathfrak{t}} \wedge \alpha$ *is consistent.*

Lemma 3.3.11 *For atoms* $\mathfrak{t}_1$ *and* $\mathfrak{t}_2$, *the following statements are equivalent.*

1. $\widehat{\mathfrak{t}_1} \wedge \langle a\rangle\widehat{\mathfrak{t}_2}$ *is consistent.*

2. $\langle\overline{a}\rangle\widehat{\mathfrak{t}_1} \wedge \widehat{\mathfrak{t}_2}$ *is consistent.*

Lemma 3.3.12 *There exist* $\mathfrak{t}_1, \ldots, \mathfrak{t}_k \in AT(\alpha_0)$ *and* $a_1, \ldots a_k \in \Sigma$ *(*$k \geq 0$*) such that* $\mathfrak{t}_k \xrightarrow{a_k}_{AT} \mathfrak{t}_{k-1} \ldots \xrightarrow{a_1}_{AT} \mathfrak{t}_0$, *where* $\mathfrak{t}_k$ *is a root atom.*

Proof: Consider the least set R containing $\mathfrak{t}_0$ and closed under the following condition: if $\mathfrak{t}_1 \in R$ and for some $a \in \Sigma$ there exists $\mathfrak{t}_2$ such that $\mathfrak{t}_2 \xrightarrow{a}_{AT} \mathfrak{t}_1$, then $\mathfrak{t}_2 \in R$. Now, if there exists an atom $\mathfrak{t}' \in R$ such that $\mathfrak{t}'$ is a root then we are done. Suppose not, then we have $\vdash \widetilde{R} \supset \neg root$. But then, we can show that $\vdash \widetilde{R} \supset [P]\widetilde{R}$. By rule *Ind-past* and above we get $\vdash \widetilde{R} \supset \boxminus\neg root$. But then $\mathfrak{t}_0 \in R$ and hence $\vdash \widehat{\mathfrak{t}_0} \supset \widetilde{R}$ and therefore we get $\vdash \widehat{\mathfrak{t}_0} \supset \boxminus\neg root$. Since $\neg\diamondminus root \in CL(\alpha_0)$ and from Lemma 3.3.10(5) we get $\neg\diamondminus root \in \mathfrak{t}_0$. From axiom (A4a) and Lemma 3.3.10(4) we have $\diamondminus root \in \mathfrak{t}_0$ which contradicts the consistency of $\mathfrak{t}_0$. □

Lemma 3.3.13 *Consider the path* $\mathfrak{t}_k \xrightarrow{a_k}_{AT} \mathfrak{t}_{k-1} \ldots \xrightarrow{a_1}_{AT} \mathfrak{t}_0$ *where* $\mathfrak{t}_k$ *is a root atom,*

1. *For all* $j \in \{0, \ldots, k-1\}$, *if* $[\overline{a}]\alpha \in \mathfrak{t}_j$ *and* $\mathfrak{t}_{j+1} \xrightarrow{a}_{AT} \mathfrak{t}_j$ *then* $\alpha \in \mathfrak{t}_{j+1}$.

2. *For all* $j \in \{0, \ldots, k-1\}$, *if* $\langle\overline{a}\rangle\alpha \in \mathfrak{t}_j$ *and* $\mathfrak{t}_{j+1} \xrightarrow{b}_{AT} \mathfrak{t}_j$ *then* $b = a$ *and* $\alpha \in \mathfrak{t}_{j+1}$.

3. *For all* $j \in \{0, \ldots, k-1\}$, *if* $\diamondminus\alpha \in \mathfrak{t}_j$ *then there exists* $i : j \leq i \leq k$ *such that* $\alpha \in \mathfrak{t}_i$.

Proof: (1) Since $\mathfrak{t}_{j+1} \xrightarrow{a}_{AT} \mathfrak{t}_j$, we have $\widehat{\mathfrak{t}_{j+1}} \wedge \langle a\rangle\widehat{\mathfrak{t}_j}$ is consistent. By lemma 3.3.11, $\widehat{\mathfrak{t}_j} \wedge \langle\overline{a}\rangle\widehat{\mathfrak{t}_{j+1}}$ is consistent, which implies $[\overline{a}]\alpha \wedge \langle\overline{a}\rangle\widehat{\mathfrak{t}_{j+1}}$ is consistent (by omitting some conjuncts). Therefore $\langle\overline{a}\rangle(\alpha \wedge \widehat{\mathfrak{t}_{j+1}})$ is consistent. Using (*NG*-) we get $\alpha \wedge \widehat{\mathfrak{t}_{j+1}}$ is consistent and since $\mathfrak{t}_{j+1}$ is an atom, we have $\alpha \in \mathfrak{t}_{j+1}$.

(2) Suppose $\mathfrak{t}_{j+1} \xrightarrow{b}_{AT} \mathfrak{t}_j$, we first show that $b = a$. Suppose this is not true, since $\mathfrak{t}_{j+1} \xrightarrow{b}_{AT} \mathfrak{t}_j$, we have $\widehat{\mathfrak{t}_j} \wedge \langle \overline{b} \rangle \widehat{\mathfrak{t}_{j+1}}$ is consistent. And therefore $\widehat{\mathfrak{t}_j} \wedge \langle \overline{b} \rangle \mathit{True}$ is consistent. From axiom (A2c), $\widehat{\mathfrak{t}_j} \wedge [\overline{a}] \mathit{False}$ is consistent. If $\langle \overline{a} \rangle \alpha \in \mathfrak{t}_j$, then we get $\langle \overline{a} \rangle \alpha \wedge [\overline{a}] \mathit{False}$ is consistent. Therefore $\langle \overline{a} \rangle (\alpha \wedge \mathit{False})$ is consistent. From (NG-) we have $\alpha \wedge \mathit{False}$ is consistent, contradicting the consistency of α (ensured by the fact that $\langle a \rangle \alpha \in \mathfrak{t}_j$ is consistent).

To show $\alpha \in \mathfrak{t}_{j+1}$ observe that $\langle \overline{a} \rangle \alpha \in \mathfrak{t}_j$ implies $[\overline{a}] \alpha \in \mathfrak{t}_j$ (by axiom (A2b) and closure condition). By previous argument we get $\alpha \in \mathfrak{t}_{j+1}$.
(3) Suppose $\Diamond\!\!\!\!-\alpha \in \mathfrak{t}_j$ and $\mathfrak{t}_{j+1} \xrightarrow{a}_{AT} \mathfrak{t}_j$. If $\alpha \in \mathfrak{t}_j$ then we are done. Else, by axiom (A4b) and the previous argument, we have $\langle \overline{a} \rangle \Diamond\!\!\!\!-\alpha \in \mathfrak{t}_j$. From (2) we have $\Diamond\!\!\!\!-\alpha \in \mathfrak{t}_{j+1}$. Continuing in this manner, we either get an i where $\alpha \in \mathfrak{t}_i$ (in which case we are done) or we get $\Diamond\!\!\!\!-\alpha \in \mathfrak{t}_k$. Since $\mathfrak{t}_k$ is the root atom, we have $\widehat{\mathfrak{t}_k} \wedge \neg \langle \mathcal{P} \rangle \mathit{True}$ is consistent. Since $\Diamond\!\!\!\!-\alpha \in \mathfrak{t}_k$, we get $\widehat{\mathfrak{t}_k} \wedge (\alpha \vee \langle \mathcal{P} \rangle \alpha)$ is consistent. Thus we have $\widehat{\mathfrak{t}_k} \wedge \alpha$ is consistent and therefore $\alpha \in \mathfrak{t}_k$. □

Lemma 3.3.14 *For all $\mathfrak{t} \in AT(\alpha_0)$ if $\sigma \rightsquigarrow_i \beta \notin \mathfrak{t}$ then there exists a path $\rho_t^{\mathfrak{t}_k} : \mathfrak{t} = \mathfrak{t}_1 \xrightarrow{a_1}_{AT} \mathfrak{t}_2 \ldots \xrightarrow{a_{k-1}}_{AT} \mathfrak{t}_k$ which conforms to σ such that one of the following conditions hold.*

- $\beta \notin \mathfrak{t}_k$.
- $\mathit{moves}(\mathfrak{t}_k) \cap \sigma(\mathfrak{t}_k) = \emptyset$.

Proof: Consider the least set R containing $\mathfrak{t}$ and closed under the following condition:

- if $\mathfrak{t}_1 \in R$ then for every transition $\mathfrak{t}_1 \xrightarrow{a}_{AT} \mathfrak{t}_2$ such that $a \in \sigma(\mathfrak{t}_1)$ we have $\mathfrak{t}_2 \in R$.

If there exists an atom $\mathfrak{t}' \in R$ such that $\beta \notin \mathfrak{t}'$ or if $\mathit{moves}(\mathfrak{t}') \cap \sigma(\mathfrak{t}') = \emptyset$, then we are done. Suppose not, then we have $\vdash \widetilde{R} \supset \beta$ and $\vdash \widetilde{R} \supset \bigvee_{a \in \Sigma} (\langle a \rangle \mathit{True} \wedge (\sigma)_i : a)$.

Claim : The following are derivable.

1. $\vdash (\widetilde{R} \wedge \mathbf{turn}_i \wedge (\sigma)_i : a) \supset [a] \widetilde{R}$.
2. $\vdash (\mathbf{turn}_{\overline{i}} \wedge \widetilde{R}) \supset [\mathcal{N}] \widetilde{R}$.

The claim can be verified as follows. To prove 1, suppose the claim does not hold. We have that $(\widetilde{R} \wedge \mathbf{turn}_i \wedge (\sigma)_i : a) \wedge \langle a \rangle \neg \widetilde{R}$ is consistent. Let $R' = AT(\alpha_0) \setminus R$.

If $R' = \emptyset$ then $R = AT(\alpha_0)$ in which case its easy to see that the claim holds. If $R' \neq \emptyset$, then we have $(\widetilde{R} \wedge \mathbf{turn}_i \wedge (\sigma)_i : a) \wedge \langle a \rangle \widetilde{R'}$ is consistent. Hence for some $\mathfrak{t}_1 \in R$ and $\mathfrak{t}_2 \in R'$, we have $(\widehat{\mathfrak{t}_1} \wedge \mathbf{turn}_i \wedge (\sigma)_i : a)) \wedge \langle a \rangle \widehat{\mathfrak{t}_2}$ is consistent. Which implies $\mathfrak{t}_1 \xrightarrow{a}_{AT} \mathfrak{t}_2$ and this transition conforms to σ. By closure condition on R, $\mathfrak{t}_2 \in R$ which gives us the required contradiction. Proof of 2 is similar.

End of claim

From the above claim and applying $(Ind \rightsquigarrow)$ rule we get $\vdash \widetilde{R} \supset \sigma \rightsquigarrow_i \beta$. But $\mathfrak{t} \in R$ and therefore $\vdash \widehat{\mathfrak{t}} \supset \sigma \rightsquigarrow_i \beta$, contradicting the assumption that $\sigma \rightsquigarrow_i \beta \notin \mathfrak{t}$. $\square$

Model construction in terms of atoms: Consider the tree structure T_{AT} which consists of the path $\mathfrak{t}_k \xrightarrow{a_k}_{AT} \mathfrak{t}_{k-1} \ldots \xrightarrow{a_1}_{AT} \mathfrak{t}_0$ and the tree unfolding of $AT(\alpha_0)$ rooted at $\mathfrak{t}_0$. By Lemma 3.3.14, the structure T_{AT} satisfies the property if there exists a node $\mathfrak{t}$ such that $\sigma \rightsquigarrow_i \beta \notin \mathfrak{t}$ then there exists a path $\mathfrak{t} = \mathfrak{t}_1 \xrightarrow{a_1}_{AT} \ldots \xrightarrow{a_m}_{AT} \mathfrak{t}_m$ such that $\beta \notin \mathfrak{t}_m$ or $moves(\mathfrak{t}_m) \cap \sigma(\mathfrak{t}_m) = \emptyset$. It would be easy if we could build a model for α_0 based on the tree structure T_{AT}. Since the size of $AT(\alpha_0)$ is exponential in the size of α_0, this would immediately lead to the decidability of the satisfiability problem of the logic as well. The trouble is that the structure $(AT(\alpha_0), \rightarrow_{AT})$ need not be deterministic. There might exist atoms $\mathfrak{t}_1, \mathfrak{t}_2, \mathfrak{t}_3$ such that $\mathfrak{t}_1 \xrightarrow{a_1}_{AT} \mathfrak{t}_2$ and $\mathfrak{t}_1 \xrightarrow{a_1}_{AT} \mathfrak{t}_3$. Thus $\rightarrow_{AT}$ does not define a parital function and therefore extracting a deterministic model out of the structure $(AT(\alpha_0), \rightarrow_{AT})$ is not straightforward.

For the purpose of showing completeness, we can circumvent this problem by constructing the canonical model in terms of maximal consistent sets. We show how this can be done below.

Canonical model construction: Let $\mathfrak{M}$ denote the set of all maximal consistent sets (MCS). We use $\mathfrak{m}, \mathfrak{m}'$ to range over MCS's. Since α_0 is consistent, there exists an MCS $\mathfrak{m}_0$ such that $\alpha_0 \in \mathfrak{m}_0$. Define a transition relation on MCS's as follows: $\mathfrak{m} \xrightarrow{a}_{\mathfrak{m}} \mathfrak{m}'$ iff $\{\langle a \rangle \alpha \mid \alpha \in \mathfrak{m}'\} \subseteq \mathfrak{m}$. It is easy to see that for any MCS $\mathfrak{m}$ we have $\mathfrak{m} \cap CL(\alpha_0) \in AT(\alpha_0)$.

We define the model M as follows. From Lemma 3.3.12 and Lemma 3.3.13 it is easy to see that there exist MCS's $\mathfrak{m}_1, \ldots, \mathfrak{m}_k \in \mathfrak{M}$ and $a_1, \ldots a_k \in \Sigma$ $(k \geq 0)$ such that $\mathfrak{m}_k \xrightarrow{a_k}_{\mathfrak{m}} \mathfrak{m}_{k-1} \ldots \xrightarrow{a_1}_{\mathfrak{m}} \mathfrak{m}_0$, where $\mathfrak{m}_j \cap CL(\alpha_0) = \mathfrak{t}_j$. Now this path defines a (finite) tree $T_0 = (S_0, \Rightarrow_0, s_0, \widehat{\lambda})$ rooted at s_0, where $S_0 = \{s_0, s_1, \ldots, s_k\}$, and for all $j \in \{0, \cdots, k\}$, s_j is labelled by the MCS $\mathfrak{m}_{k-j}$. The relation $\Rightarrow_0$ is defined in

the obvious manner. From now we will simply say $\alpha \in s$ where s is the tree node, to mean that $\alpha \in \mathfrak{m}$ where $\mathfrak{m}$ is the MCS associated with node s. The turn function is defined as expected: $\widehat{\lambda}(s_j) = i$ if $\mathbf{turn}_i \in s_j$ else $\widehat{\lambda}(s_j) = \overline{i}$.

Inductively assume that we have a tree $T_k = (S_k, \Rightarrow_k, s_0, \widehat{\lambda}_k)$ such that the past formulas at every node have "witnesses" as above. Pick a node $s \in S_k$ such that $\langle a \rangle True \in s$ but there is no $s' \in S_k$ such that $s \stackrel{a}{\Rightarrow} s'$. Now, if $\mathfrak{m}$ is the MCS associated with node s, there exists an MCS $\mathfrak{m}'$ such that $\mathfrak{m} \stackrel{a}{\rightarrow}_{\mathfrak{m}} \mathfrak{m}'$. Pick a new node $s' \notin S_k$ and define $T_{k+1} = (S_{k+1}, \Rightarrow_{k+1}, s_0, \widehat{\lambda}_k)$ where $S_{k+1} = S_k \cup \{s'\}$ and $\Rightarrow_{k+1} = \Rightarrow_k \cup \{(s, a, s')\}$, where $\mathfrak{m}'$ is the MCS associated with s'. It is easy to see that every node in T_{k+1} has witnesses for past formulas as well. The turn function is extended as defined earlier for the newly added nodes.

Now consider $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ defined by: $S = \bigcup_{k \geq 0} S_k$ and $\Rightarrow = \bigcup_{k \geq 0} \Rightarrow_k$. Define the model $M = (T, V)$ where $V(s) = w \cap P$, where w is the MCS associated with s. Let $\Rightarrow^*$ denote the reflexive and transitive closure of $\Rightarrow$ relation.

Lemma 3.3.15 *For any $s \in S$, we have the following properties.*

1. *If $[a]\alpha \in s$ and $s \stackrel{a}{\Rightarrow} s'$ then $\alpha \in s'$.*
2. *If $\langle a \rangle \alpha \in s$ then there exists s' such that $s \stackrel{a}{\Rightarrow} s'$ and $\alpha \in s'$.*
3. *If $[\overline{a}]\alpha \in s$ and $s' \stackrel{a}{\Rightarrow} s$ then $\alpha \in s'$.*
4. *If $\langle \overline{a} \rangle \alpha \in s$ then there exists s' such that $s' \stackrel{a}{\Rightarrow} s$ and $\alpha \in s'$.*
5. *If $\boxminus \alpha \in s$ and $s' \Rightarrow^* s$ then $\alpha \in s'$.*
6. *If $\diamondminus \alpha \in s$ then there exists s' such that $s' \Rightarrow^* s$ and $\alpha \in s'$.*

Proof: Cases (1) to (5) can be shown using standard modal logic techniques. (6) follows from the existense of a root atom (Lemma 3.3.12) and axiom (A4b). □

Lemma 3.3.16 *For all $\psi \in Past(P)$, for all $s \in S$, $\psi \in s$ iff $\rho^s_{s_0}, s \models \psi$.*

Proof: This follows from Lemma 3.3.15 using an inductive argument. □

Lemma 3.3.17 *For all i, for all $\sigma \in Strat^i(P^i)$, for all $c \in \Sigma$, for all $s \in S$, $(\sigma)_i : c \in s$ iff $c \in \sigma(s)$.*

Proof: The proof is by induction on the structure of σ.
$\sigma = [\psi \mapsto a]^i$: Suppose $([\psi \mapsto a]^i)_i : c \in s$. If $c = a$ then the claim holds trivially. If $c \neq a$ then from (A5a) we get that $\neg\psi \in s$. From Lemma 3.3.16 we have $\rho^s_{s_0}, s \not\models \psi$. Therefore by definition we have $[\psi \mapsto a]^i(s) = \Sigma$ and $c \in \sigma(w)$.

Conversely, suppose $([\psi \mapsto a]^i)_i : c \notin s$. From (A5a) we have $a \neq c$. From (A5b) we get $\psi \in s$. By Lemma 3.3.16 $\rho^s_{s_0}, s \models \psi$. Therefore $c \notin \sigma(s)$ by definition.
The cases when $\sigma = (\sigma_1 + \sigma_2)$ and $\sigma = (\sigma_1 \cdot \sigma_2)$ follow easily from induction hypothesis.
$\sigma = \pi \Rightarrow \sigma'$: Let $\rho^s_{s_0} : s_0 \stackrel{a_0}{\Rightarrow} \cdots \stackrel{a_{k-1}}{\Rightarrow} s_k = s$ be the unique path from the root to s. Suppose $(\pi \Rightarrow \sigma')_i : c \in s$. To show $c \in (\pi \Rightarrow \sigma')(s)$. Suffices to show that $\rho^s_{s_0}$ conforms to π implies $c \in \sigma'(s)$. From (A6c) we have $conf_\pi \supset (\sigma')_i : c \in s$. Recall that $conf_\pi = \boxdot(\langle\overline{a}\rangle\mathbf{turn}_{\overline{\imath}} \supset \langle\overline{a}\rangle(\pi)_{\overline{\imath}} : a)$ denotes that all opponent moves in the past conform to π. Thus we get $\diamondminus(\langle\overline{a}\rangle\mathbf{turn}_{\overline{\imath}} \wedge [\overline{a}](\neg(\pi)_{\overline{\imath}} : a)) \vee (\sigma')_i : c \in s$. We have two cases,

- if $(\sigma')_i : c \in s$ then by induction hypothesis we get $c \in \sigma'(s)$. Therefore by definition $c \in (\pi \Rightarrow \sigma)(s)$.

- otherwise, we have $\diamondminus(\langle\overline{a}\rangle\mathbf{turn}_{\overline{\imath}} \wedge [\overline{a}](\neg(\pi)_{\overline{\imath}} : a)) \in s$. From Lemma 3.3.15(6), there exists $s_l \in \rho^s_{s_0}$ such that $\langle\overline{a}\rangle\mathbf{turn}_{\overline{\imath}} \wedge [\overline{a}](\neg(\pi)_{\overline{\imath}} : a) \in s_l$. By Lemma 3.3.15(4) there exists $s_{l-1} \in \rho^s_{s_0}$ such that $s_{l-1} \stackrel{a}{\Rightarrow} s_l$. From Lemma 3.3.15(3), $\neg(\pi)_{\overline{\imath}} : a \in s_{l-1}$. Since s_{l-1} is an MCS, we have $(\pi)_{\overline{\imath}} : a \notin s_{l-1}$. By induction hypothesis, $a \notin \pi(s_{l-1})$, therefore we have that $\rho^s_{s_0}$ does not conform to π.

($\Leftarrow$) Conversely, suppose $(\pi \Rightarrow \sigma')_i : c \notin s$, to show $c \notin (\pi \Rightarrow \sigma')(s)$. It suffices to show that $\rho^s_{s_0}$ conforms to π and $c \notin \sigma'(s)$. From axiom (A6c), we have $conf_\pi \wedge \neg(\sigma)_i : c \in s$. Rewriting this we get $(\boxdot(\langle\overline{a}\rangle\mathbf{turn}_{\overline{\imath}} \supset \langle\overline{a}\rangle(\pi)_{\overline{\imath}} : a)) \wedge (\neg(\sigma)_i : c) \in s$. From the first conjunct and using Lemma 3.3.15 we get $\rho^s_{s_0}$ conforms to π. The second conjunct implies $(\sigma)_i : c \notin s$ and by induction hypothesis we get $c \notin \sigma'(s)$. Thus by definition $c \notin (\pi \Rightarrow \sigma)(s)$. □

Lemma 3.3.18 *For all $\alpha \in \Pi$, for all $s \in S$, $\alpha \in s$ iff $M, s \models \alpha$.*

Proof: The proof is by induction on the structure of α. The non-trivial cases are as follows.
$\alpha = (\sigma)_i : c$. From Lemma 3.3.17 we have $(\sigma)_i : c \in s$ iff $c \in \sigma(s)$ iff by semantics $M, s \models (\sigma)_i : c$.

$\alpha = \sigma \rightsquigarrow_i \beta$.
($\Rightarrow$) We show the following:

1. If $\sigma \rightsquigarrow_i \beta \in s$ and there exists a transition $s \overset{a}{\Rightarrow} s'$ such that $a \in \sigma(s)$, then $\{\beta, \sigma \rightsquigarrow_i \beta\} \subseteq s'$. Suppose $\sigma \rightsquigarrow_i \beta \in s$, from (A7) we have $\beta \in s$. We have two cases to consider.

   - $\widehat{\lambda}(s) = i$: We have $\mathbf{turn}_i \in s$. Since $a \in \sigma(s)$, by Lemma 3.3.17 we have $(\sigma)_i : a \in s$. From (A7) we get $[a](\sigma \rightsquigarrow_i \beta) \in s$. By Lemma 3.3.15(1) we have $\sigma \rightsquigarrow_i \beta \in s'$.
   - $\widehat{\lambda}(s) = \overline{i}$: We have $\mathbf{turn}_{\overline{i}} \in s$. From (A7) we get $[\mathcal{N}](\sigma \rightsquigarrow_i \beta) \in s$, since s is an MCS we have for every $a \in \Sigma$, $[a](\sigma \rightsquigarrow_i \beta) \in s$. By Lemma 3.3.15(1) we have $\sigma \rightsquigarrow_i \beta \in s'$.

   By applying (A7) at s' we get $\beta \in s'$.

2. If $\sigma \rightsquigarrow_i \beta \in s$ then there exists s' such that $s \overset{a}{\Rightarrow} s'$ and $a \in \sigma(s)$. From axiom (A7), $\bigvee_{a \in \Sigma}(\langle a \rangle \mathit{True} \wedge (\sigma)_i : a) \in s$. Since s is an MCS, there exists an a such that $\langle a \rangle \mathit{True} \wedge (\sigma)_i : a \in s$. By Lemma 3.3.15(2), there exists an s' such that $s \overset{a}{\Rightarrow} s'$ and by Lemma 3.3.17 $a \in \sigma(s)$.

(1) ensures that whenever $\sigma \rightsquigarrow_i \beta \in s$ and there exists a path $\rho_s^{s_k}$ which conforms to σ, then we have $\{\beta, \sigma \rightsquigarrow_i \beta\} \subseteq s_k$. Since $\beta \in \mathit{Past}(P)$, by Lemma 3.3.16 we have $M, s_k \models \beta$. (2) ensures that for all paths $\rho_s^{s_k}$ which conforms to σ, $\mathit{moves}(s_k) \cap \sigma(s_k) \neq \emptyset$. Therefore we get $M, s \models \sigma \rightsquigarrow_i \beta$.
($\Leftarrow$) Conversely suppose $\sigma \rightsquigarrow_i \beta \notin s$, to show $M, s \not\models \sigma \rightsquigarrow_i \beta$. It suffices to show that there exists a path $\rho_s^{s_k}$ that conforms to σ such that $M, s_k \not\models \beta$ or $\mathit{moves}(s_k) \cap \sigma(s_k) = \emptyset$.

Let $\mathfrak{t} = s \cap \mathit{CL}(\alpha_0)$, we have $\mathfrak{t} \in \mathit{AT}(\alpha_0)$ and $\sigma \rightsquigarrow_i \beta \in \mathfrak{t}$. By Lemma 3.3.14, there exists a path in the atom graph $\mathfrak{t} = \mathfrak{t}_1 \overset{a_1}{\rightarrow}_{AT} \mathfrak{t}_2 \ldots \overset{a_k}{\rightarrow}_{AT} \mathfrak{t}_k$ such that $\beta \notin \mathfrak{t}_k$ or $\mathit{moves}(\mathfrak{t}_k) \cap \sigma(\mathfrak{t}_k) = \emptyset$. $\mathfrak{t}_1$ can be extended to the MCS s. Let $\mathfrak{t}_2' = \mathfrak{t}_2 \cup \{\alpha | [a_1]\alpha \in s\}$. Its easy to check that $\mathfrak{t}_2'$ is consistent. Consider any MCS s_2 extending $\mathfrak{t}_2'$, we have $s \overset{a_1}{\Rightarrow} s_2$. Continuing in this manner we get a path in $s = s_1 \overset{a_1}{\Rightarrow} s_2 \ldots \overset{a_{k-1}}{\Rightarrow} s_k$ in M which conforms to σ where either $\beta \notin s_k$ or $\mathit{moves}(s_k) \cap \sigma(s_k) = \emptyset$. □

This leads us to the following theorem which asserts that the axiom system is complete.

Theorem 3.3.19 *For all formula α_0, if α_0 is consistent then α_0 is satisfiable.*

Proof: Suppose α_0 is a consistent formula, then $\{\alpha_0\}$ can be extended to a maximal consistent set $\mathfrak{m}_0$. By the construction of the model $M = (T, V)$, there exists a node s in T such that s is labelled with $\mathfrak{m}_0$. By Lemma 3.3.18, $M, s \models \alpha_0$ and therefore α_0 is satisfiable. □

3.3.6 Truth Checking

The truth checking problem for the logic is stated as: given a model M and a formula α_0, determine whether $M, s_0 \models \alpha_0$. In this section we show that the truth checking problem for the logic is decidable. Models of the logic were defined to be extensive form game trees extended with a valuation function.

However for algorithmic analysis, we need to present the infinite tree in some finite fashion. As shown in the previous chapter we can think of the infinite extensive form game tree being generated by the tree unfolding of a finite game arena. Thus a model M can be presented as a structure $(W, \rightarrow, w_0, \lambda, V)$ where $(W, \rightarrow, w_0, \lambda)$ constitutes a game arena and $V : W \rightarrow 2^{P'}$ is a valuation function where P' is a finite subset of P. This can also be thought of as a Kripke structure extended with a turn function λ. Formulas of the logic are then interpreted on the structure T_M which is the extensive form game tree resulting from the tree unfolding of M.

We can now rephrase the truth checking problem in the following manner: given a model in terms of a Kripke structure M and a formula α_0, determine whether $T_M, s_0 \models \alpha_0$. The idea is to build a tree automaton which accepts T_M iff $T_M, s_0 \models \alpha_0$. Since T_M is a possibly infinite structure, we need to consider automata running over infinite trees. Towards this objective, we formally define a Büchi tree automaton below.

Büchi tree automata: A tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ is said to be a k-ary tree if for all $s \in S$, the out degree of s is k, i.e. $\mid \vec{s} \mid = k$.

Definition 3.3.20 *A Büchi tree automaton running over k-ary trees is a structure $\mathcal{T} = (\mathbb{Q}, \mathbb{R}, \mathbb{I}, \mathbb{F})$ where*

- *$\mathbb{Q}$ is the set of states*
- *$\mathbb{R} \subseteq \mathbb{Q} \times S \times \mathbb{Q}^k$ is the transition relation.*

- $\mathbb{I} \subseteq \mathbb{Q}$ *is the set of initial states.*
- $\mathbb{F} \subseteq \mathbb{Q}$ *is the set of "good" states.*

A run of $\mathcal{T}$ on T is a $\mathbb{Q}$ labelled tree $\mathfrak{R} = (S, \Rightarrow, s_0, \widehat{\lambda}, l)$ where $l : S \to \mathbb{Q}$ is the labelling function defined as,

- $l(s_0) \in \mathbb{I}$.
- For any s we have $(l(s), s, l(s_1), \ldots, l(s_k)) \in \mathbb{R}$.

For an infinite sequence of states $\varphi : q_0 q_1 \ldots$, let $Inf(\varphi)$ denote the set of states occuring infinitely often in φ. For a path $\rho = s_0 a_0 s_1 \ldots$, let $l(\rho) = l(s_0) l(s_1) \ldots$. A run $\mathfrak{R}$ is accepted by $\mathcal{T}$ if for all paths ρ in $\mathfrak{R}$, $Inf(l(\rho)) \cap \mathbb{F} \neq \emptyset$.

Proposition 3.3.21 ([VW86]) *The emptiness problem of Büchi tree automata can be solved in polynomial time.*

We make use of the following definition and preliminary results for the construction.

Definition 3.3.22 *For a strategy specification $\sigma \in Strat^i(P^i)$ let $\mathcal{A}_\sigma = (Q_\sigma, \delta_\sigma, o_\sigma, I_\sigma)$ be the advice automaton corresponding to σ. We construct a deterministic automaton $\mathfrak{A}_\sigma$ as follows: $\mathfrak{A}_\sigma = (\mathfrak{Q}_\sigma, \boldsymbol{\delta}_\sigma, \mathfrak{o}_\sigma, \mathfrak{I}_\sigma)$ where*

- $\mathfrak{Q}_\sigma = 2^{Q_\sigma}$.
- $\boldsymbol{\delta}_\sigma(X, w, a) = \{q' \mid \exists q \in X \text{ with } q' \in \delta_\sigma(q, w, a)\}$.
- $\mathfrak{o}_\sigma : \mathfrak{Q}_\sigma \times W \to 2^\Sigma$ *which satisfies the condition: for all $X = \{q_1, \ldots, q_k\} \in \mathfrak{Q}_\sigma$ and all game positions w, $\mathfrak{o}_\sigma(X, w) = \{o_\sigma(q_1, w), \ldots, o_\sigma(q_k, w)\}$.*
- $\mathfrak{I}_\sigma = I_\sigma$.

The automaton $\mathfrak{A}_\sigma$ is constructed by performing the standard subset construction on the underlying structure of the advice automaton $\mathcal{A}_\sigma$ and in which the output symbols are aggregated.

The run of $\mathfrak{A}_\sigma$ on a tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$ is the $\mathfrak{Q}_\sigma$ labelled tree $\mathfrak{R} = (S, \Rightarrow, s_0, \widehat{\lambda}, l)$ where $l : S \to \mathfrak{Q}_\sigma$ is the labelling function defined as,

- $l(s_0) = \mathfrak{I}_\sigma$.

- For any s_k where $s_k \stackrel{a}{\Rightarrow} s'_k$ we have $l(s'_k) = \boldsymbol{\delta}_\sigma(l(s_k), s_k, a)$.

For a deterministic automaton $\mathfrak{A}_\sigma$ and tree $T = (S, \Rightarrow, s_0, \widehat{\lambda})$, let $\mathfrak{R}(\mathfrak{A}_\sigma, T) = (S, \Rightarrow, s_0, \widehat{\lambda}, l_{\mathfrak{A}_\sigma})$ denote the run of $\mathfrak{A}_\sigma$ on T. The following proposition, which can be easily verified, asserts the correctness of the subset construction.

Proposition 3.3.23 *For all $\sigma \in Strat^i(P^i)$, for all $T = (S, \Rightarrow, s_0, \widehat{\lambda})$, for all $s \in S$ and for all $q \in Q_\sigma$,*

- *$q \in l_{\mathfrak{A}_\sigma}(s)$ iff there exists a run $\mathfrak{R}(\mathcal{A}_\sigma, T) = (S, \Rightarrow, s_0, \widehat{\lambda}, l_{\mathcal{A}_\sigma})$ of $\mathcal{A}_\sigma$ on T such that $l_{\mathcal{A}_\sigma}(s) = q$.*

Lemma 3.3.24 *For all $\sigma \in Strat^i(P^i)$, for all $T = (S, \Rightarrow, s_0, \widehat{\lambda})$, for all $s_k \in S$ such that $\widehat{\lambda}(s) = i$, we have $\sigma(s) = \mathfrak{o}_\sigma(l_{\mathfrak{A}_\sigma}(s), last(s))$.*

Proof: The proof is by induction on the structure of σ.
$\sigma = [\psi \mapsto a]^i$ - If $\rho^{s_k}_{s_0}, s_k \models \psi$ then for all runs $\mathfrak{R}(\mathcal{A}_\sigma, T) = (S, \Rightarrow, s_0, \widehat{\lambda}, l_{\mathcal{A}_\sigma})$ of $\mathcal{A}_\sigma$ on T we have $o_\sigma(l_{\mathcal{A}_\sigma}(s_k), last(s_k)) = a$. By Proposition 3.3.23 we get for all $q \in l_{\mathfrak{A}_\sigma}(s_k)$, $o_\sigma(q, last(s)) = a$ and therefore we have $\mathfrak{o}_\sigma(l_{\mathfrak{A}_\sigma}(s), last(s)) = \{a\}$.

If $\rho^{s_k}_{s_0}, s_k \not\models \psi$ then for all $b \in \Sigma$, there exists a run $\mathfrak{R}(\mathcal{A}_\sigma, T) = (S, \Rightarrow, s_0, \widehat{\lambda}, l_{\mathcal{A}_\sigma})$ of $\mathcal{A}_\sigma$ on T such that $o_\sigma(l_{\mathcal{A}_\sigma}(s_k), last(s_k)) = b$. Again by applying Proposition 3.3.23 we can deduce that $\mathfrak{o}_\sigma(l_{\mathfrak{A}_\sigma}(s_k), last(s_k)) = \Sigma$.
For the cases when $\sigma = \sigma_1 + \sigma_2$ and $\sigma = \sigma_1 \cdot \sigma_2$ the claim easily follows by the construction of the advice automaton and the induction hypothesis.
$\sigma = \pi \Rightarrow \sigma_1$ - Suppose $\rho^{s_k}_{s_0} = s_0 a_0 \dots a_{k-1} s_k$ conforms to π, i.e. for all j with $0 \leq j < k$, $a_j \in \pi(s_j)$, then from the semantics we have $(\pi \Rightarrow \sigma_1)(s_k) = (\sigma_1)(s_k)$. In this case, the output of $\mathcal{A}_{\pi \Rightarrow \sigma_1}$ at node s_k is same as the output of $\mathcal{A}_{\sigma_1}$ at s_k. Thus by construction of $\mathfrak{A}_\sigma$ we have $\mathfrak{o}_{\pi \Rightarrow \sigma_1}(l_{\mathfrak{A}_\sigma}(s_k), last(s_k)) = \mathfrak{o}_{\sigma_1}(l_{\mathfrak{A}_\sigma}(s_k), last(s_k))$. By induction hypothesis we have $\mathfrak{o}_{\sigma_1}(l_{\mathfrak{A}_\sigma}(s_k), last(s_k)) = \sigma_1(s_k)$. Thus we get $\mathfrak{o}_{\pi \Rightarrow \sigma_1}(l_{\mathfrak{A}_\sigma}(s_k), last(s_k)) = \sigma_1(s_k) = (\pi \Rightarrow \sigma_1)(s_k)$.

Suppose $\rho^{s_k}_{s_0}$ does not conform to π then by semantics $(\pi \Rightarrow \sigma_1)(s) = \Sigma$. In this case, the advice automaton moves to a state q_{free} where it is free to produce any output. From the construction of $\mathfrak{A}_\sigma$ we can deduce that $\mathfrak{o}_{\pi \Rightarrow \sigma_1}(l_{\mathfrak{A}_\sigma}(s_k), last(s_k)) = \Sigma$. □

Let $AT(\alpha_0)$ denote the set of all atoms of α_0 (see Definition 3.3.9). Let $\mathfrak{t}_0 = \{\mathfrak{t} \in AT(\alpha_0) \mid \Diamond\alpha \in \mathfrak{t}$ implies $\alpha \in \mathfrak{t}\}$. For $\mathfrak{t}, \mathfrak{t}' \in AT(\alpha_0)$, define $\mathfrak{t} \xrightarrow{a} \mathfrak{t}'$ iff the following conditions hold.

- For all $[\bar{a}]\alpha \in CL(\alpha_0)$, $[\bar{a}]\alpha \in \mathfrak{t}'$ iff $\alpha \in \mathfrak{t}$.
- For all $[a]\alpha \in CL(\alpha_0)$, if $[a]\alpha \in \mathfrak{t}$ then $\alpha \in \mathfrak{t}'$.

Tree automaton construction: Given a model M and a formula α_0, the objective is to construct a Büchi tree automaton $\mathcal{T}(M, \alpha_0)$ running over k-ary trees for some fixed $k \in \mathbb{N}$. However, the branching degree of nodes in T_M need not be uniform. We first observe that since the automaton construction is parameterised by the model M it is possible to always normalise the model to one where all nodes have the same branching degree. Let $\mathfrak{b} = \max_{w \in W} | \vec{w} |$, i.e. $\mathfrak{b}$ denotes the maximum out degree of positions in W. We can "normalise" M by adding edges to dummy nodes so that all nodes are of out degree $\mathfrak{b}$. These newly added dummy nodes are sink nodes of the graph, i.e. all the out going edges are self loops. A tree automaton running on such a normalised tree would enter an accept state on encountering a dummy node and remain in this state. Thus any path ending in a dummy node is disregarded by the automaton and only paths which were present in M are analysed for consistency requirements.

Therefore without loss of generality we can assume that the model M satisfies the condition: for all $w \in W$, $| \vec{w} | = \mathfrak{b}$. Thus the tree T_M is a $\mathfrak{b}$-ary tree.

Let $\mathfrak{S}(\alpha_0) = \{\sigma_1, \ldots, \sigma_m\}$ be the strategy specifications appearing in α_0 and for $1 \leq j \leq m$ and $\mathfrak{A}_{\sigma_j} = (\mathfrak{Q}_{\sigma_j}, \boldsymbol{\delta}_{\sigma_j}, \mathfrak{o}_{\sigma_j}, \mathfrak{I}_{\sigma_j})$ be the deterministic automaton corresponding to the advice automaton $\mathcal{A}_{\sigma_j}$. Let $\mathfrak{Q} = \mathfrak{Q}_{\sigma_1} \times \ldots \times \mathfrak{Q}_{\sigma_m}$, we use $\mathbf{X} = (X_1, \ldots, X_m)$ to denote elements of $\mathfrak{Q}$.

Intuitively the tree automaton works as follows: It keeps track of the maximal consistent subsets (atoms) of α_0 and simulates the advice automata $\mathfrak{A}_{\sigma_1}, \ldots, \mathfrak{A}_{\sigma_m}$ in parallel. At a game position s, for a subformula $(\sigma)_i : a$ in the atom, it ensures that the action "a" is a possible output of the advice automaton $\mathcal{A}_\mu$. However, $\neg(\sigma \rightsquigarrow_i \beta)$ is a requirement which says that there exists a game position where $enabled_\sigma$ does not hold or β is false. We keep track of such formulas in a "requirement set" U. When the tree automaton branches, it guesses for each branch which requirements need to be satisfied on that particular branch. The Büchi acceptance condition is simply all those states where the "requirement set" U is empty.

Formally we have $\mathcal{T}(M, \alpha_0) = (\mathbb{Q}, \mathbb{R}, \mathbb{I}, \mathbb{F})$ where the set of states $\mathbb{Q} = (AT(\alpha_0) \cup \{\checkmark, \ltimes\}) \times (2^{CL(\alpha_0)})^3 \times \mathfrak{Q}$ satisfies the property: for all $((\mathfrak{t}, d), U, Z, Y, \mathbf{X}) \in \mathbb{Q}$, if $d = \checkmark$ then

- for all $\langle a \rangle \alpha \in Z$ it is the case that $\alpha \in \mathfrak{t}$.

- for all $(\sigma_j)_i : a \in CL(\alpha_0)$, $(\sigma_j)_i : a \in \mathfrak{t}$ iff $a \in \mathfrak{o}_{\sigma_j}(X_j, last(s))$.

We also assume the existence of a special *accept* state in $\mathbb{Q}$. The set of initial states $\mathbb{I}$ satisfies the condition: $((\mathfrak{t}, \checkmark), U, Z, Y, \mathbf{X}^0) \in \mathbb{I}$ iff

- $\mathfrak{t} \in \mathfrak{t}_0$, $U = \emptyset$ and $Z = \emptyset$.
- $Y = \{\langle a \rangle \alpha \in CL(\alpha_0) \mid \langle a \rangle \alpha \in \mathfrak{t}\}$.
- $\mathbf{X}^0 = (X_1^0, \ldots, X_m^0)$.

The sets Z and Y are used to keep track of the $\langle a \rangle \alpha$ formulas and ensure that the edge relation is consistent with these formulas. For a game position s, let $last(s) = w$ and let $\vec{w} = \{w_1, \ldots, w_{\mathfrak{b}}\}$ with $w \xrightarrow{a_j} w_j$ for $1 \leq j \leq \mathfrak{b}$. The automaton at a state $((\mathfrak{t}, d), U, Z, Y, \mathbf{X})$ reading the game position s, guesses a partition of $U = U_1 \cup \ldots \cup U_{\mathfrak{b}}$. Note that to be technically precise, it needs to be mentioned that the structure M is also encoded into the state space of the automaton. However, to avoid cluttering of the notations we do not explicitly represent this. The transition relation $\mathbb{R}$ is defined as follows.
$\langle (((\mathfrak{t}_1, d_1), U_1', Z_1, Y_1, \mathbf{X}^1), a_1), \ldots, (((\mathfrak{t}_{\mathfrak{b}}, d_{\mathfrak{b}}), U_{\mathfrak{b}}', Z_{\mathfrak{b}}, Y_{\mathfrak{b}}, \mathbf{X}^k), a_{\mathfrak{b}}) \rangle \in \mathbb{R}(((\mathfrak{t}, d), U, Z, Y, \mathbf{X}), s)$ iff the following conditions hold.

- If $d = \ltimes$ then $d_j = \ltimes$ for all $j : 1 \leq j \leq \mathfrak{b}$.
- For $V(s) = \mathfrak{t} \cap Voc(\alpha_0)$ and for all $j : 1 \leq j \leq \mathfrak{b}$, $\mathfrak{t} \xrightarrow{a_j} \mathfrak{t}_j$.
- $Z_j = \{\langle a \rangle \alpha \in Y \mid a = a_j\}$ and $Z_1, \ldots, Z_{\mathfrak{b}}$ form a partition of Z.
- $U_j' = \begin{cases} \{\neg(\sigma \rightsquigarrow_i \beta) \in U_j \mid \beta, enabled_\sigma \in C_j\} & \text{if } U \neq \emptyset \\ \{\neg(\sigma \rightsquigarrow_i \beta) \in C_j \mid \beta, enabled_\sigma \in C_j\} & \text{if } U = \emptyset \end{cases}$
- $Y_j = \{\langle a \rangle \alpha \mid \langle a \rangle \alpha \in C_j\}$.
- For $1 \leq j \leq \mathfrak{b}$, $1 \leq r \leq m$, $X_r^j = \boldsymbol{\delta}_{\sigma_r}(X_r, s, a_j)$.

The state with entry $\ltimes$ corresponds to a reject state and once the automaton enters the rejects state it remains in that state for all transitions. The Büchi acceptance condition is, $F = \{((\mathfrak{t}, \checkmark), U, Z, Y, \mathbf{X}) \in \mathbb{Q} \mid U = \emptyset\} \cup \{accept\}$.

Theorem 3.3.25 *Given a model M and a formula α_0 we have $T_M, s_0 \models \alpha_0$ iff $Lang(\mathcal{T}(M, \alpha_0)) \neq \emptyset$.*

Proof: ($\Rightarrow$): Suppose $T_M, s_0 \models \alpha$, we show that there exists an accepting run of $\mathcal{T}(M, \alpha_0)$ on T_M. For a node s, let $\mathfrak{t}_s = \{\alpha \in CL(\alpha_0) \mid M, s \models \alpha\}$. We define the labelling $l_\mathcal{T}$ inductively as follows. $l_\mathcal{T}(s_0) = ((\mathfrak{t}_{s_0}, \checkmark), \emptyset, \emptyset, Y, \mathbf{X}^0)$ where $Y = \{\langle a\rangle\alpha \in CL(\alpha_0) \mid \langle a\rangle\alpha \in \mathfrak{t}\}$ and $\mathbf{X}^0 = (X_1^0, \ldots, X_m^0)$.

Now consider any node s such that $l_\mathcal{T}(s)$ is defined and the labelling function is not defined on its successors. Let $l_\mathcal{T}(s) = ((\mathfrak{t}, \checkmark), U, Z, Y, \mathbf{X})$ and let $\{s_1, \ldots, s_\mathfrak{b}\}$ be the successor nodes of s with $s \overset{a_j}{\Rightarrow} s_j$ for all $j : 1 \leq j \leq \mathfrak{b}$. If $U \neq \emptyset$, for each formula $\neg(\sigma \rightsquigarrow_i \beta) \in U$ we have $T_M, s \models \neg(\sigma \rightsquigarrow_i \beta)$. Thus there exists a path $s = \mathfrak{s}_1\mathfrak{s}_2 \ldots \mathfrak{s}_q$ such that $T_M, \mathfrak{s}_q \models \neg\beta \vee \neg enabled_\sigma$ and for all intermediate node $\mathfrak{s}_r$, $T_M, \mathfrak{s}_r \models \beta \wedge enabled_\sigma$. We also have $\mathfrak{s}_2 \in \{s_1, \ldots, s_\mathfrak{b}\}$. Set U_j to be the set of all formulas $\neg(\sigma \rightsquigarrow_i \beta) \in U$ such that $\mathfrak{s}_2 = s_j$. Now for all $j : 1 \leq j \leq \mathfrak{b}$, $l_\mathcal{T}(s_j) = ((\mathfrak{t}_{s_j}, \checkmark), U_j, Z_j, Y_j, \mathbf{X}^j)$ where

- $Z_j = \{\langle a_j\rangle\alpha \in CL(\alpha_0) \mid \langle a_j\rangle\alpha \in Y\}$.
- $Y_j = \{\langle a\rangle\alpha \in CL(\alpha_0) \mid \langle a\rangle\alpha \in \mathfrak{t}_{s_j}\}$.
- For all $r : 1 \leq r \leq m$, $X_r^j = \delta_r^\mathcal{D}(X_r, s, a_j)$.

It is straight forward to verify that this defines a valid run of the automaton and that it is accepting.
($\Leftarrow$): Suppose $Lang(\mathcal{T}(M, \alpha_0)) \neq \emptyset$, let $T_M = (S, \Rightarrow, s_0, \widehat{\lambda})$ and $\mathfrak{R} = (S, \Rightarrow, s_0, \widehat{\lambda}, l_\mathcal{T})$ be the accepting run of $\mathcal{T}(M, \alpha_0)$ on T_M. The labelling function $l_\mathcal{T}$ labels nodes of the run tree with states of the automaton. Thus for a game position s, $l_\mathcal{T}(s)$ is a tuple $((\mathfrak{t}, d), U, Z, Y, \mathbf{X})$. We denote by $l_\mathcal{T}(s)[j]$ the jth component of the tuple with $l_\mathcal{T}(s)[1] = (\mathfrak{t}, d)$. For a formula α, we also write $\alpha \in l_\mathcal{T}(s)$ to mean $\alpha \in \mathfrak{t}$. We show the following:

- for all $\alpha \in CL(\alpha_0)$ and for all $s \in S$, $\alpha \in l_\mathcal{T}(s)$ iff $M, s \models \alpha$.

The proof is by induction on the structure of α and the interesting cases are as follows.
$\alpha = p$ - We have $M, s \models p$ iff $p \in V(s)$ iff $p \in l_\mathcal{T}(s)$.
$\alpha = (\sigma)_i : a$ - Recall that $\mathfrak{S}(\alpha_0) = \{\sigma_1, \ldots, \sigma_m\}$. Let r be the index of σ in $\mathfrak{S}(\alpha_0)$, i.e. $\sigma_r = \sigma$. Let $l_\mathcal{T}(s) = ((\mathfrak{t}, d), U, Z, Y, \mathbf{X})$ where $\mathbf{X} = (X_1, \ldots, X_m)$. $M, s \models (\sigma)_i : a$ iff $a \in \sigma(s)$ iff $a \in \mathfrak{o}_{\sigma_r}(X_r, last(s))$ (from lemma 3.3.24). Since $\mathfrak{R}$ is an accepting run, we have $d = \checkmark$ and by definition of the automaton we have $a \in O_{\sigma_r}(X_r, s)$ iff $(\sigma)_i : a \in l_\mathcal{T}(s)$.

$\alpha = \sigma \rightsquigarrow_i \beta$ - Suppose $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s)$ we need to show that $T_M, s \models \sigma \rightsquigarrow_i \beta$. It suffices to show that for every path $\rho_s^{s_k}$ which conforms to σ, $T_M, s_k \models \beta$ and $\overrightarrow{s_k} \cap \sigma(s_k) \neq \emptyset$. We show the following:

1. If $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s)$ and there exists an s' such that $s \stackrel{a}{\Rightarrow} s'$ with $a \in \sigma(s)$ then $\{\beta, \sigma \rightsquigarrow_i \beta\} \subseteq l_{\mathcal{T}}(s')$. Let r be the index of σ in $\mathfrak{S}(\alpha_0)$, i.e. $\sigma_r = \sigma$. Let $l_{\mathcal{T}}(s) = ((\mathfrak{t}, d), U, Z, Y, \mathbf{X})$ where $\mathbf{X} = (X_1, \ldots, X_m)$. Suppose $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s)$ by definition of atom we have $\beta \in l_{\mathcal{T}}(s)$. The following two cases arise.
   - $\widehat{\lambda}(s) = i$: We have $\mathbf{turn}_i \in l_{\mathcal{T}}(s)$. Since $a \in \sigma(s)$, by lemma 3.3.24 we have $a \in \mathfrak{o}_{\sigma_r}(X_r, last(s))$. By definition of state we have $(\sigma)_i : a \in l_{\mathcal{T}}(s)$. From definition of atom we have $[a](\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$. By definition of the transition relation we have $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s')$ and by definition of atom $\beta \in l_{\mathcal{T}}(s')$.
   - $\widehat{\lambda}(s) = \overline{i}$: We have $\mathbf{turn}_{\overline{i}} \in l_{\mathcal{T}}(s)$. From definition of atom we get $[\mathcal{N}](\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$ and thus we have for every $a \in \Sigma$, $[a](\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$. By definition of the transition relation we have $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s')$ and by definition of atom $\beta \in l_{\mathcal{T}}(s')$.

2. If $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s)$ then there exists s' such that $s \stackrel{a}{\Rightarrow} s'$ and $a \in \sigma(s)$. Suppose $\sigma \rightsquigarrow_i \beta \in l_{\mathcal{T}}(s)$, by definition of atom we have $enabled_\sigma \in l_{\mathcal{T}}(s)$. By expanding he abbreviation we get $\bigvee_{a \in \Sigma}(\langle a \rangle True \wedge (\sigma)_i : a) \in l_{\mathcal{T}}(s)$. By definition of atom we get, there exists an $a \in \Sigma$ such that $\langle a \rangle True$, $(\sigma)_i : a \in l_{\mathcal{T}}(s)$. Since $\mathfrak{R}$ is an accepting run of $\mathcal{T}(M, \alpha_0)$, we have there exists an s' such that $s \stackrel{a}{\Rightarrow} s'$ and $a \in O_\sigma(l_{\mathcal{T}}(s), s)$. By lemma 3.3.24 we have $a \in \sigma(s)$.

(1) ensures that whenever $\sigma \rightsquigarrow_i \beta \in s$ and there exists a path $\rho_s^{s_k}$ which conforms to σ, then we have $\{\beta, \sigma \rightsquigarrow_i \beta\} \subseteq l_{\mathcal{T}}(s_k)$. Applying induction hypothesis we get $T_M, s_k \models \beta$. (2) ensures that for all paths $\rho_s^{s_k}$ which conforms to σ, $\overrightarrow{s_k} \cap \sigma(s_k) \neq \emptyset$. It suffices to show that for every path $\rho_s^{s_k}$ which conforms to σ, $T_M, s_k \models \beta$ and $\overrightarrow{s_k} \cap \sigma(s_k) \neq \emptyset$. From the semantics, we get $T_M, s \models \sigma \rightsquigarrow_i \beta$.

Conversely, suppose $\sigma \rightsquigarrow_i \beta \notin l_{\mathcal{T}}(s)$. By definition of atom, $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$. We have the following cases:

1. if $\beta \notin l_{\mathcal{T}}(s)$ then applying induction hypothesis we get $T_M, s \not\models \beta$ and from semantics we get $T_M, s \not\models \sigma \rightsquigarrow_i \beta$.

2. if $enabled_\sigma \notin l_{\mathcal{T}}(s)$ then by expanding the abbreviation we get $\bigvee_{a \in \Sigma}(\langle a \rangle True \wedge (\sigma)_i : a) \notin l_{\mathcal{T}}(s)$. Thus for every $a \in \Sigma$, either $\langle a \rangle True \notin l_{\mathcal{T}}(s)$ or $(\sigma)_i : a \notin l_{\mathcal{T}}(s)$.

Let $Y = \{a_1, \ldots, a_k\} \subseteq \Sigma$ such that $\forall j : 0 \leq j \leq k$, $\langle a_j \rangle True \in l_{\mathcal{T}}(s)$. Thus for all j, there exists s_j such that $s \stackrel{a_j}{\Rightarrow} s_j$. Let r be the index of σ in $\mathfrak{S}(\alpha_0)$ and let $l_{\mathcal{T}}(s) = ((\mathfrak{t}, d), U, Z, Y, \mathbf{X})$ where $\mathbf{X} = (X_1, \ldots, X_m)$. For each $a_j \in Y$, we have $(\sigma)_i : a_j \notin l_{\mathcal{T}}(s)$ and thus by definition of $l_{\mathcal{T}}(s)$, $a_j \notin \mathfrak{o}_{\sigma_r}(X_r, last(s))$. By lemma 3.3.24 we have $a_j \notin \sigma_r(s)$. Thus we can deduce that $moves(s) \cap \sigma(s) = \emptyset$. From semantics we have $T_M, s \not\models \sigma \rightsquigarrow_i \beta$.

3. if $\langle \mathcal{N} \rangle \neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$ and β, $enabled_\sigma \notin l_{\mathcal{T}}(s)$ then we have the following cases:

   - suppose $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)[2]$ (recall that $l_{\mathcal{T}}(s)[2]$ denotes the "requirement set" of the tree automaton). Since $\mathfrak{R}$ is accepting for all paths starting at s, eventually the requirement set becomes empty. Thus we get a path $s = s_1 s_2 \ldots s_k$ such that $\forall j : 1 \leq j < k$, $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_j)[2]$ and $\neg(\sigma \rightsquigarrow_i \beta) \notin l_{\mathcal{T}}(s_k)[2]$. By definition of the transition relation of the automaton either $\beta \notin l_{\mathcal{T}}(s_k)$ or $enabled_\sigma \notin l_{\mathcal{T}}(s_k)$. Applying induction hypothesis and using the semantics we can conclude that $T_M, s \not\models \sigma \rightsquigarrow_i \beta$.
   - suppose $\neg(\sigma \rightsquigarrow_i \beta) \notin l_{\mathcal{T}}(s)[2]$. We show that there exists a path $s = s_1 s_2 \ldots s_k$ such that for all $j : 1 \leq j \leq k$, $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_j)$ and one of the following holds:
     - $\beta \notin l_{\mathcal{T}}(s_k)$ or $enabled_\sigma \notin l_{\mathcal{T}}(s_k)$.
     - $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_k)[2]$

     Since $\langle \mathcal{N} \rangle \neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s)$, $\exists s_2$ such that $s \stackrel{a}{\Rightarrow} s_2$ and $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_2)$. If β or $enabled_\sigma \in l_{\mathcal{T}}(s_2)$ or if $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_k)[2]$ then we are done. Otherwise, by definition of atom we get $\langle \mathcal{N} \rangle \neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_2)$. By repeating the argument there exists s_3 such that $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_3)$. Since $\mathfrak{R}$ is an accepting run, for every path starting at s the requirement set eventually becomes empty. Thus we get a path $s = s_1 s_2 \ldots s_{k-1} s_k$ such that $l_{\mathcal{T}}(s_{k-1})[2] = \emptyset$ and for all $j : 1 \leq j \leq k$, $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_j)$ and $\beta, enabled_\sigma \notin l_{\mathcal{T}}(s_k)$. By definition of the transition function of the automaton we have $\neg(\sigma \rightsquigarrow_i \beta) \in l_{\mathcal{T}}(s_k)[2]$.

□

Complexity of truth checking: For the given formula α_0, let $|\alpha_0|$ denote the size of α_0. The states of the tree automaton are the atoms of α_0 and the states of the determinised versions of the advice automata. The model M is also hard-coded into the automaton structure. Since the number of strategy specifications occurring in α_0 is bounded by $|\alpha_0|$, the size of the tree automaton is doubly exponential in $|\alpha_0|$ and linear in $|M|$. To solve the truth checking problem, we need to check the emptiness of $\mathcal{T}(M, \alpha_0)$. Since $\mathcal{T}(M, \alpha_0)$ is a Büchi tree automaton, this can be done in polynomial time (due to Proposition 3.3.21). Thus we get a total time complexity of doubly exponential in $|\alpha_0|$ and polynomial in the size of the model.

3.3.7 Extension to joint actions

We can also use strategy specifications to reason about joint actions of players in the setting of multi-player games. For this purpose, in this section we assume that the set of players $N = \{1, \ldots, n\}$. For each player $i \in N$, let Σ_i be the set of actions of player i and $\Sigma = \bigcup_{i \in N} \Sigma_i$.

For an action $a \in \Sigma$, let $loc(a) = \{i \in N \mid a \in \Sigma_i\}$. Intuitively $loc(a)$ denotes the set of all players for which a is a joint action. For a subset of players $\mathcal{C} \subseteq N$, let $\overline{\mathcal{C}}$ denote the set $\overline{\mathcal{C}} = N \setminus \mathcal{C}$. Consider the following syntax for strategy specifications: for each $\mathcal{C} \subseteq N$ we have

$$Strat^{\mathcal{C}} := [\psi \mapsto a]^{\mathcal{C}} \mid \sigma_1 + \sigma_2 \mid \sigma_1 \cdot \sigma_2 \mid \pi \Rightarrow \sigma$$

where $loc(a) = \mathcal{C}$ and $\pi \in Strat^{\overline{\mathcal{C}}}$.

For the construct $[\psi \mapsto a]^{\mathcal{C}}$, the action a needs to be a joint action of the subset of players $\mathcal{C}$ and it says that for all game positions where the condition ψ holds, the players in $\mathcal{C}$ play the joint action a. $\pi \Rightarrow \sigma$ says that in the past if the observable behaviour of the coalition $\overline{\mathcal{C}}$ conforms to π then $\mathcal{C}$ stick to the specification σ. The game structures under consideration now need not be turn based and can allow concurrent moves by a subset of players. The notion of strategies of players can be defined appropriately. Embedding these strategy specifications into a modal logic as given in Section 3.3 enables us to reason about the abilities of subsets of players and mechanisms by which they can ensure specific outcomes.

However, note that what is done here is different from the concept of coalitions studied in classical game theory. Cooperative games [Jon80], which is a widely studied branch of game theory, reasons about the rational behaviour of collections of players.

Unlike non-cooperative games (the kind of games we have looked at so far), in cooperative games players have the freedom of pre-play communication and to make binding agreements. Very often the literature also talks about side payments where players are allowed to transfer utility even before the game is played. In contrast, in our setting players do not communicate and therefore negotiations of players are never part of the model. Even though the assertions involve subsets of players, we are still reasoning about abilities of a subset $\mathcal{C}$ with respect to its compliment set $\overline{\mathcal{C}}$.

The alternating temporal logic (ATL) [AHK02] also reasons about joint strategies of players and how they can ensure certain outcomes. The main construct in ATL has the form $\langle\langle \mathcal{C} \rangle\rangle \alpha$ which says that the subset of players in $\mathcal{C}$ has a strategy to ensure α. ATL formulas are interpreted over concurrent game structures which are game arenas where concurrent moves of players are enabled. In its original form, ATL talks about "existence of strategies" of players. Naming functional (memoryless) strategies and including them explicitly into the logical language was suggested in [WvdHW07] and [vdHJW05]. ATL extended with the ability to specify actions of players in the formulas was studied in [Ågo06] and [Bor07]. Since the unfolding of the game structure encodes past information, the logic itself can be extended with past modalities as well as knowledge modalities in order to reason about the history information and epistemic conditions used in strategizing by players ([JvdH04],[vdHW02]).

Reasoning about abilities of coalitions as done in ATL can be achieved in our framework by replacing each joint action a with $loc(a)$, the subset of players which share the action a. The interpretation would then be that the subset of players in $loc(a)$ have a joint action to ensure a particular outcome. For a finite game arena, memoryless strategies can be easily coded up in terms of strategy specifications by using special propositions to identify each game position uniquely. Thus reasoning about functional (memoryless) strategies explicitly in the framework can also be achieved. The modal logic which embeds the specification language can always be extended with the past, future and knowledge modalities depending on the application in mind. However, in the context of specifications to help build advice functions for players, it is not clear what joint actions (without communication) achieve in a game theoretic sense.

3.3.8 Discussion

The approach we adopted to specify strategies is close in spirit to reasoning about games in the dynamic logic framework. As mentioned earlier, the proposed logic

can talk about "existence of strategies" with the assertion of the form $null^i \rightsquigarrow_i \beta$. However, it can make stronger assertions of the form $\sigma \rightsquigarrow_i \beta$ which also specifies the mechanism which need to be used by player i in order to guarantee the outcome β.

Reasoning in finite extensive form games: The logic can also be used to perform outcome based reasoning in finite extensive form games. For $i \in N$, let Θ_i denote the set of propositions coding the utility of players. Consider the following formula:

$$BT^i(\sigma, \sigma') = \bigwedge_{\theta_i \in \Theta_i} (\sigma' \rightsquigarrow_i (\mathbf{leaf} \supset \theta_i) \supset \sigma \rightsquigarrow_i (\mathbf{leaf} \supset \theta_i))$$

This says that irrespective of what player $\overline{i}$ plays if there exists a strategy μ' satisfying σ' such that θ_i is guaranteed, then there also exists a strategy μ satisfying σ which guarantees θ_i. In other words, for player i, the specification σ is better than σ'.

Given a finite set of strategy specifications Υ^i for player i, we say that σ is the best strategy if the following holds:

$$Best^i(\sigma) = \bigwedge_{\sigma' \in \Upsilon^i} BT^i(\sigma, \sigma')$$

Representing complete strategies: Note that in the case of finite extensive form games, we can code up the game positions uniquely using propositions. In this case, it is possible to represent complete strategies in terms of strategy specifications as well. Suppose the proposition $p_i^1, \ldots p_i^k$ uniquely identifies all player i game positions, the specification representing a complete strategy would have the form $\sigma = [p_i^1 \mapsto a_1]^i \cdots [p_i^k \mapsto a_k]^i$. In this particular scenario, the notion of strategy comparison and best strategy reduces to the classical notions by taking the set Υ^i to be the set of all strategies for player i.

This also shows that dynamic logic based reasoning about games as mentioned in Section 2.4 is subsumed by the proposed logical framework. However, the most important aspect in which we deviate from dynamic logic based reasoning is that we do not require that the logic codes up complete strategies of players. Thus formulas of the logic need not be parameterised by the specific game structure under consideration, in particular it need not depend on the specific bound on the length of plays. This also enables us to reason about more "generic" games of unbounded duration.

Chapter 4

Dynamic logic on game composition

In the previous chapter we looked at logical analysis of strategies where the game representation itself is assumed to be atomic. The emphasis was to reason about strategies and develop a framework to build strategies in a compositional manner as in the case of programs. The logical formalism developed does not explicitly dictate the structure of the game arena and thus is closer in spirit to endogenously defined logics.

Game theoretic techniques have often been used in dealing with various issues in logic. In the latter half of the last century, the language of games was extensively used to discuss questions in model theory, typically in the context of model construction and for comparing models or expressiveness of logical languages. This was demonstrated by the work of Hintikka [Hin68] whereby the meaning of quantifiers was explained in game theoretic terms. In this approach, semantics of quantified first order formulas is given in terms of a two player zero sum game. Logical notions like satisfiability are reduced to existence of winning strategies for one of the players. Games have also found useful applications in model theory as illustrated by the characterisation of elementary equivalence in terms of Ehrenfeucht-Fraisse games [Ehr61]. The question here is whether two structures are distinguishable with respect to first order formulas. This question can be translated into a game where the winning strategy of a player corresponds to a separating formula.

The notion of composition is inherent to any logical formalism, thus games which model logical properties can be thought of as being composed in some structural fashion. When games are themselves structured, players strategic response reflects this structure as well. For games of bounded length, an action labelled modal logic similar to the one presented in Section 2.4 reflects game and strategy structure

well, but when we consider unbounded play as arising from unbounded repetition of games, the situation is different. Propositional game logic [Par85], the seminal work on logical aspects of game theory, builds composite structure into games. Game logic initiated the study of game structure using algebraic properties. Pauly [Pau01] has built on this to describe abilities of coalitions to achieve desired goals and to provide interesting relationships between programs and games.

The strategies used by a player in such a composite game would depend on not just the outcome specification but also what strategy was used, especially by opponents, in the past. The history information can then be analysed by taking into account the underlying structure of the composite game. We suggest that in reasoning about structured games, it is useful for the strategies of players to also reflect the structure. Thus rather than reasoning about the strategies in the composed game, one should look at strategies in atomic games and compose such atomic game strategy pairs.

In this chapter, we look at a framework where both games and strategies are structurally built and where the structure is explicitly represented in the formulas of the logic. In the case of extensive form games, we suggest that considering game - strategy pairs is useful: suppose that we have a 2-stage game g_1 followed by g_2. Consider player 1 strategizing at the end of g_1, when g_2 is about to start; her planning depends not only on how g_2 is structured, but also how her opponent had played in g_1. Thus her strategizing in the composite game $g_1; g_2$ is best described as follows: consider g_1 in extensive form as a tree, and the subtree obtained by opponent employing π; when g_2 starts from any of the leaf nodes of this subtree, play according to σ. We encode this as $(g_1, \pi); (g_2, \sigma)$, and see (g_2, σ) as a response to (g_1, π). Thus the "programs" of this logic are game - strategy pairs of this kind.

We consider a propositional dynamic logic, the programs of which are regular expressions over atomic pairs of the form (g, σ) where g is a finite game tree in extensive form, and σ is a strategy specification, structured syntactically. The central syntactic device consists of interactive structure in strategies and algebraic structure not only on games but on game - strategy pairs. While the technical result is a complete axiomatization and the decidability of the satisfiability problem, we see this contribution as an advocacy of studying algebraic structure on strategies, induced by that on games.

In contrast, in normal form games strategies are presented in an abstract manner and the reasoning in such games are driven by outcome specifications. A normal

form game can be viewed as an extensive form game abstracted into a tree of depth one, where edges are labelled by a tuple of strategies, one for each player. There is no past and future that strategies refer to, and we only speak of notions like rational response, dominant strategies, equilibrium and so on. However, when we consider repeated games, or games composed of smaller games, the notion of strategic response of a player to other players' moves becomes relevant, pretty much in the same way as it is used in extensive form games. History information, as well as epistemic attitudes of players become relevant. In this setting, we consider composition of game play pairs, corresponding to the fact that the reasoning performed in single stage is outcome based. On the technical front, a complete axiomatization of the logic can be provided as in the case of extensive form games. However, the central objective is to highlight the logical differences between composition of normal form games and that of extensive form games, in terms of the reasoning involved.

In the setting of compositional games, to illustrate the difference between reasoning about games and reasoning about strategies, we find it instructive to first review propositional game logic.

4.1 Game logic

Propositional game logic [Par85] initiated the logical study of determined two person zero sum games in a compositional framework.

Syntax

Let the two players be denoted as player 1 and player 2. As in the case of propositional dynamic logic, the language of game logic consists of two sorts: games and propositions. Let Γ_0 be a set of atomic games and P a set of atomic propositions. The set of composite games Γ and the set of formulas Φ is built from the following syntax:

$$\begin{aligned} \Gamma &:= g \mid \gamma_1;\gamma_2 \mid \gamma_1 \cup \gamma_2 \mid \gamma^* \mid \gamma^d \\ \Phi &:= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\gamma\rangle\varphi \end{aligned}$$

where $p \in P$ and $g \in \Gamma_0$. Let $[\gamma]\varphi := \neg\langle\gamma\rangle\neg\varphi$ and $\gamma_1 \cap \gamma_2 := (\gamma_1^d \cup \gamma_2^d)^d$.

The formula $\langle\gamma\rangle\varphi$ asserts that player 1 has a strategy in game γ to ensure φ and $[\gamma]\varphi$ expresses that player 1 does not have a strategy to ensure $\neg\varphi$, which by

determinacy is equivalent to the assertion that player 2 has a strategy to ensure φ. The intuitive definition of composite games is as follows: $\gamma_1;\gamma_2$ is the game where γ_1 is played first followed by γ_2, $\gamma_1 \cup \gamma_2$ is the game where player 1 moves first and decides whether to play γ_1 or γ_2 and then the chosen game is played. In the iterated game γ^*, player 1 can choose how often to play γ (possibly zero times). He need not declare in advance how many times γ needs to be played, but is required to eventually stop. The dual game γ^d is the same as playing the game γ with the roles interchanged.

The semantics of the logic is defined in terms of neighbourhood relations. At the atomic level, the game structure itself is not important since the emphasis is on reasoning about powers of players. Thus composition of games in effect corresponds to composing abilities of players. The formal semantics is given below.

Semantics

A game model $M = ((S, \{E_g \mid g \in \Gamma_0\}), V)$ where S is a set of states, $V : P \to 2^S$ is the valuation function and $E_g : S \to 2^{2^S}$ is a collection of *effectivity functions* which are monotonic, i.e. $X \in E_g(s)$ and $X \subseteq X'$ imply $X' \in E_g(s)$. The idea is that $X \in E_g(s)$ holds whenever player 1 has a strategy in game g to achieve X.

The truth of a formula φ in a model M at a state s (denoted $M, s \models \varphi$) is defined as follows:

$$\begin{array}{lll} M, s \models p & \text{iff} & s \in V(p) \\ M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\ M, s \models \varphi_1 \wedge \varphi_2 & \text{iff} & M, s \models \varphi_1 \text{ or } M, s \models \varphi_2 \\ M, s \models \langle\gamma\rangle\varphi & \text{iff} & \varphi^M \in E_\gamma(s) \end{array}$$

where $\varphi^M = \{s \in S \mid M, s \models \varphi\}$. The effectivity function E_γ is defined inductively for non-atomic games as follows. Let $E_g(Y) = \{s \in S \mid Y \in E_g(s) \text{ for } g \in \Gamma_0\}$. Then

$$\begin{array}{rcl} E_{\gamma_1;\gamma_2}(Y) & = & E_{\gamma_1}(E_{\gamma_2}(Y)) \\ E_{\gamma_1\cup\gamma_2}(Y) & = & E_{\gamma_1}(Y) \cup E_{\gamma_2}(Y) \\ E_{\gamma^d}(Y) & = & \overline{E_\gamma(\overline{Y})} \\ E_{\gamma^*}(Y) & = & \mu X.Y \cup E_\gamma(X) \end{array}$$

where μ denotes the least fixpoint operator. It can be shown that the monotonicity of E_γ is preserved under the game operations and therefore the least fixpoint

$\mu X.Y \cup E_\gamma(X)$ always exists.

On the technical front, the following theorems state that the satisfiability problem and model checking problem for game logic is decidable.

Theorem 4.1.1 ([Par85, Pau01]) *The satisfiability problem for game logic is EXPTIME-complete.*

Theorem 4.1.2 ([Ber05]) *The model checking problem for game logic over finite structures is in NP ∩ Co-NP.*

Expressive power

To investigate the expressive power of game logic with respect to established formalism, it is convenient to interpret game logics on Kripke structures. In this case the atomic games are one player games and all the interactive aspect is controlled syntactically using the dual operator. Over Kripke structures, in the absence of dual operator, game logic is nothing but propositional dynamic logic (PDL). The ability of PDL to express properties of programs is limited, for instance it well known that the property of well-foundedness or equivalently that of total correctness cannot be expressed in PDL. Several extensions of PDL have been proposed to get over this limitation. These include adding explicit predicates like loop [HP78] and repeat [Str81]. The dual operator increases the expressive power of game logic significantly. The property of total correctness can be expressed by making use of the dual operator and thus we get that game logic is strictly more powerful than PDL [Par85]. In fact, it can be shown that game logic is strictly more powerful than the extension of PDL with looping operator [Ber05].

Propositional μ-calculus [Koz83] was one of the most powerful logics proposed to deal with limitations of PDL. It was shown that game logic interpreted over Kripke structures can be embedded into the two variable fragment of propositional μ-calculus [Ber05]. Since the variable hierarchy of μ-calculus is strict [Ber05] we can also conclude that μ-calculus is strictly more powerful than game logic. It is quite conceivable that the model checking problem for game logic is easier than that of μ-calculus. However, Berwanger [Ber03] shows that this is not the case.

One of the main open problems in game logic is to give a complete axiomatization of valid formulas of the logic. Parikh in [Par85] proposed an axiom system and conjectured it to be complete, unfortunately no proof of this has been given so far. A complete axiomatization for the dual free fragment of game logic is presented

in [Par85]. For the iteration free fragment, a complete axiomatization is given in [Pau01].

In game logic, starting with simple atomic games, one can construct large complex games using operators like composition and union. Due to the Box-Diamond duality $\langle\gamma\rangle\varphi = \neg[\gamma]\neg\varphi$, it is easy to see that by definition, the games constructed remain determined. The compositional syntax of game logic presents an algebra for game construction. Rather than look at arbitrarily large games, this approach gives us a way of systematically studying complex games in a structured manner and to also look at their algebraic properties. One should however note that the reasoning performed in game logics is based on "existence of strategies".

Reasoning *in* compositional games

Game logic makes assertions about composing neighbourhoods or abilities of players. Players' abilities in a game arise due to the strategies they have access to. Under the assumption that players are perfectly rational, reasoning about existence of strategies suffices since such players will always be able to employ their best possible choice of actions. However, as noted earlier in the introduction, in many practical situations players have limited computational resources. In such cases, it makes sense to explicitly reason about the mechanisms by which these abilities arise in a logical framework. To reason about strategies in a compositional framework, the abstract presentation of games is not sufficient, the analysis will depend on the exact representation under consideration. We consider the two standard representations: extensive form games and normal form games and look at how a dynamic logic framework capable of reasoning about strategies can be developed.

4.2 Extensive form games

We use the notion of finite extensive form games as introduced in Section 2.1. The logical formalism (which we introduce shortly) allows formulas to explicitly refer to the game tree under consideration. We therefore need to provide a syntactic representation of the (semantic) game tree. A simple syntactic structure for specifying finite extensive form game trees is presented below.

4.2.1 Syntax for extensive form game trees

Let *Nodes* be a countable set. The syntax for specifying finite extensive form game trees given by:

$$\mathbb{G}(Nodes) := (i, x) \mid \Sigma_{a_m \in J}((i, x), a_m, t_{a_m})$$

where $i \in N$, $x \in Nodes$, $J \subseteq \Sigma$, and $t_{a_m} \in \mathbb{G}(Nodes)$.

Definition 4.2.1 *Given $g \in \mathbb{G}(Nodes)$ we define the tree T_g generated by g inductively as follows.*

- *$g = (i, x)$: $T_g = (S_g, \Rightarrow_g, \widehat{\lambda}_g, s_x)$ where $S_g = \{s_x\}$, $\widehat{\lambda}_g(s_x) = i$.*
- *$g = ((i, x), a_1, t_{a_1}) + \cdots + ((i, x), a_k, t_{a_k})$: Inductively we have trees $T_1, \ldots T_k$ where for $j : 1 \leq j \leq k$, $T_j = (S_j, \Rightarrow_j, \widehat{\lambda}_j, s_{j,0})$. Define $T_g = (S_g, \Rightarrow_g, \widehat{\lambda}_g, s_x)$ where*
  - *$S_g = \{s_x\} \cup S_{T_1} \cup \ldots \cup S_{T_k}$.*
  - *$\widehat{\lambda}_g(s_x) = i$ and for all j, for all $s \in S_{T_j}$, $\widehat{\lambda}_g(s) = \widehat{\lambda}_j(s)$.*
  - *$\Rightarrow_g = \bigcup_{j:1 \leq j \leq k}(\{(s_x, a_j, s_{j,0})\} \cup \Rightarrow_j)$.*

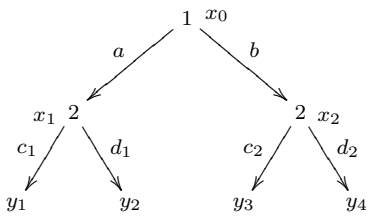


Figure 4.1: Extensive form game tree

Example 4.2.2 Consider the extensive form game tree shown in figure 4.1. The nodes are labelled with turns of players and edges with the actions. The syntactic representation of this tree can be given as follows:

- $g = ((1, x_0), a, t_1) + ((1, x_0), b, t_2)$ where
  - $t_1 = ((2, x_1), c_1, (2, y_1)) + ((2, x_1), d_1, (2, y_2))$ and
  - $t_2 = ((2, x_2), c_2, (2, y_3)) + ((2, x_2), d_2, (2, y_4))$.

□

4.2.2 The logic

The logic is a simple dynamic logic where we take regular expressions over game-strategy pairs as programs in the logic. Atomic programs are of the form (g, σ) where g is a finite extensive form game and σ is a strategy specification as defined in section 3.1. Formulas of the logic can be used to specify the result of a player following a particular strategy in a specified game enabled at a game position.

Syntax

The syntax of the logic is given by:

$$\Phi := p \in P \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle\xi\rangle^{\forall}\alpha$$

where $\xi \in \Gamma$, the set Γ consists of game strategy pairs which is defined below.

The construct ξ represents regular expressions over game-strategy pairs (g, σ). For the atomic construct (g, σ) the intuitive meaning of the formula $\langle g, \sigma\rangle^{\forall}\alpha$ is: in game g player i has a strategy μ conforming to the specification σ such that α holds at all leaf nodes reached by following μ. In other words, the strategy μ ensures the outcome α.

Game strategy pairs: The syntax for composition of game strategy specification pairs is given by,

$$\Gamma := (g, \sigma) \mid \xi_1; \xi_2 \mid \xi_1 \cup \xi_2 \mid \xi^*$$

where $g \in \mathbb{G}(\mathit{Nodes})$, $\sigma \in \mathit{Strat}^i(P^i)$.

The atomic construct (g, σ) specifies that in game g a strategy conforming to specification σ is employed. Game strategy pairs are then composed using standard dynamic logic connectives. $\xi_1 + \xi_2$ would mean playing ξ_1 or ξ_2. Sequencing in our setting does not mean the usual relational composition of games. Rather, it is the composition of game strategy pairs of the form $(g_1, \sigma_1); (g_2, \sigma_2)$. This is where the extensive form game tree interpretation makes the main difference. Since the strategy specifications are intended to be partial, a pair (g, σ) gives rise to a set of finite trees and therefore composition over trees need to be performed. ξ^* is the iteration of the ';' operator.

Remark: We use the syntax for strategy specifications given in Section 3.1. However, for technical convenience, in the case of atomic strategy specifications we restrict our attention to boolean preconditions rather than allowing past time tense

logic formulas. The main technical result of this chapter is a complete axiomatization of the logic presented above. It is easy to verify that this result can be extended to the framework which allows past time preconditions as well. The axiom system needs only to be enriched with the appropriate axioms for the past time modalities. The emphasis of this chapter is to show how compositional structure of games can be effectively used in strategizing and for this purpose boolean preconditions suffice.

For a countable set of propositions P, the set of boolean formulas over P is given by the syntax:

$$\Psi(P) := p \in P \mid \neg\psi \mid \psi_1 \vee \psi_2.$$

Given a valuation function, the semantics is defined in the obvious manner. In what follows we take atomic strategy specifications to be of the form $[\psi \mapsto a]^i$ where $\psi \in \Psi(P^i)$.

Model

The formulas of the logic express properties about game trees and strategies which are composed using tree regular expressions. These formulas are interpreted on game positions and they assert properties of the frontier nodes of the game tree. The structure of the game tree itself is dictated by the game strategy pairs. Thus models of the logic are game trees, but due to the presence of unbounded iteration, this can potentially be an infinite set of finite game trees. Alternatively, we can think of these game trees as being obtained from unfoldings of a Kripke structure. As we will see later, the logic cannot distinguish between these two. The logic introduced in Section 3.3, considered the game arena as atomic and did not reason about compositional structure within the arena. Thus we required the model to be deterministic. In the dynamic logic introduced here, the language is allowed to dictate the compositional structure of the game and the eventual game structure under consideration is specified by the formula. Therefore, we do not need to restrict ourselves to deterministic models.

A model $M = (W, \rightarrow, \lambda, V)$ where W is the set of states (or game positions), the relation $\rightarrow \subseteq W \times \Sigma \times W$, the valuation function $V : W \rightarrow 2^P$ and $\lambda : W \rightarrow N$ is a player labelling function. Note that we do not require the transition relation itself to be deterministic. However, we require that the move relation is consistent with respect to player labelling. Thus the turn function λ is required to satisfy the property:

- For all $w \in W$, if $w \xrightarrow{a} w'$ and $\lambda(w') = i$ then for all w'' such that $w \xrightarrow{a} w''$, we have $\lambda(w'') = i$.

Semantics

The truth of a formula $\alpha \in \Phi$ in a model M and a position w (denoted $M, w \models \alpha$) is defined as follows:

- $M, w \models p$ iff $p \in V(w)$.
- $M, w \models \neg\alpha$ iff $M, w \not\models \alpha$.
- $M, w \models \alpha_1 \vee \alpha_2$ iff $M, w \models \alpha_1$ or $M, w \models \alpha_2$.
- $M, w \models \langle\xi\rangle^{\forall}\alpha$ iff $\exists (w, X) \in R_\xi$ such that $\forall w' \in X$ we have $M, w' \models \alpha$.

A formula α is satisfiable if there exists a model M and a state w such that $M, w \models \alpha$.

For $\xi \in \Gamma$, we want $R_\xi \subseteq W \times 2^W$. To define the relation formally, let us first assume that R is defined for the atomic case, namely when $\xi = (g, \sigma)$. The semantics for composite game strategy pairs is given as follows:

- $R_{\xi_1;\xi_2} = \{(u, X) \mid \exists Y \subseteq W$ such that $(u, Y) \in R_{\xi_1}$ and $\forall v \in Y$ there exists $X_v \subseteq X$ such that $(v, X_v) \in R_{\xi_2}$ and $\bigcup_{v \in Y} X_v = X\}$.
- $R_{\xi_1 \cup \xi_2} = R_{\xi_1} \cup R_{\xi_2}$.
- $R_{\xi^*} = \bigcup_{n \geq 0} (R_\xi)^n$ where $(R_\xi)^n$ denotes the n-fold relational composition.

In the atomic case when $\xi = (g, \sigma)$ we want a pair (u, X) to be in R_ξ if the game g is enabled at state u and there is a strategy conforming to the specification σ such that X is the set of leaf nodes of the strategy. In order to make this precise, we will require the following notations and definitions.

Restriction on trees: For $w \in W$, let T_w denote the tree unfolding of M starting at w. We say the game g is enabled at a state w if the structure g can be embedded in T_w with respect to the enabled actions and player labelling. Since M need not be deterministic, there could be multiple embeddings, and therefore we work with the maximal embedding (denoted $T_w \restriction g$) and this is the game tree under consideration. Formally this can be defined as follows:

Given a state w and $g \in \mathbb{G}(Nodes)$, let $T_w = (S^w_M, \Rightarrow_M, \widehat{\lambda}_M, s_w)$ and $T_g = (S_g, \Rightarrow_g, \widehat{\lambda}_g, s_{g,0})$. The restriction of T_w with respect to the game g (denoted $T_w \restriction g$) is the subtree of T_w which is generated by the structure specified by T_g. The restriction is defined inductively as follows: $T_w \restriction g = (S, \Rightarrow, \widehat{\lambda}, s_0, f)$ where $f : S \to S_g$. Initially $S = \{s_w\}$, $\widehat{\lambda}(s_w) = \widehat{\lambda}_M(s_w)$, $s_0 = s_w$ and $f(s_w) = s_{g,0}$.

For any $s \in S$, let $f(s) = t \in S_g$. Let $\{a_1, \ldots, a_k\}$ be the outgoing edges of t, i.e. for all $j : 1 \leq j \leq k$, $t \stackrel{a_j}{\Rightarrow}_g t_j$. For each a_j, let $\{s^1_j, \ldots, s^m_j\}$ be the nodes in S^w_M such that $s \stackrel{a_j}{\Rightarrow}_M s^l_j$ for all $l : 1 \leq l \leq m$. Add nodes $s^1_j, \ldots, s^m_j$ to S and the edges $s \stackrel{a_j}{\Rightarrow} s^l_j$ for all $l : 1 \leq l \leq m$. Also set $\widehat{\lambda}(s^l_j) = \widehat{\lambda}_M(s^l_j)$ and $f(s^l_j) = t_j$.

We say that a game g is enabled at w (denoted $enabled(g, w)$) if the tree $T_w \restriction g = (S, \Rightarrow, \widehat{\lambda}, s_0, f)$ satisfies the following property: for all $s \in S$,

- $\vec{s} = \overrightarrow{f(s)}$,
- if $\vec{s} \neq \emptyset$ then $\widehat{\lambda}(s) = \widehat{\lambda}_g(f(s))$.

A strategy for player i on $T_w \restriction g$ can still be thought of as a subtree where at every player i vertex, there is exactly one outgoing edge and for player $\overline{i}$ vertices, all outgoing edges are included. In the functional notion, this corresponds to picking not just an action but also a successor node on the action.

For a game tree T, let $\Omega^i(T)$ denote the set of strategies of player i on the game tree T and $frontier(T)$ denote the set of all leaf nodes of T.

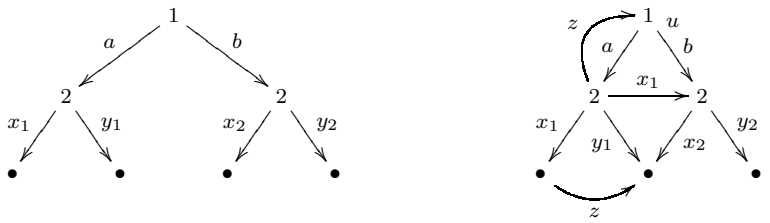
Atomic game-strategy pair: For an atomic game-strategy pair $\xi = (g, \sigma)$ we define R_ξ as follows:
Let g be the game with a single node $g = (i, x)$

- $R_{(g,\sigma)} = \{(u, \{u\})\}$ if $enabled(g, u)$ holds, for all $i \in N$, for all $\sigma \in Strat^i(P^i)$.

For $g = ((i, x), a_1, t_{a_1} + \ldots + (i, x), a_k, t_{a_k})$

- $R_{(g,\sigma)} = \{(u, X) \mid enabled(g, u)$ and $\exists \mu \in \Omega^i(T_u \restriction g)$ such that $\mu \models_i \sigma$ and $frontier(\mu) = X\}$.

Example 4.2.3 Let the extensive form game g be the one given in Figure 4.2(a) and the Kripke structure M be as shown in Figure 4.2(b). For the node u of the structure the restriction $T_u \restriction g$ is shown in Figure 4.3. This is the maximal subtree of T_u according to the structure dictated by g. For instance at node v_1 there are



(a): T_g (b): T_w

Figure 4.2: Extensive form game tree and Kripke structure M

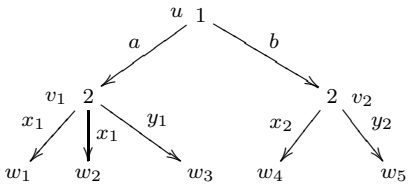


Figure 4.3: Restriction of T_w to T_g at u

two x_1 labelled edges present in M and therefore both have to be included in $T_u \restriction g$ as well.

Now consider the player 1 strategy specification $\sigma = null^1$. At node u, the choice "a" can ensure player 1 the states $\{w_1, w_2, w_3\}$ and the choice "b" can ensure the states $\{w_4, w_5\}$. Therefore we have the relation $R_{(g,\sigma)} = \{(u, \{w_1, w_2, w_3\}), (u, \{w_4, w_5\}), (v_1, \{w_1, w_2, w_3\}), (v_2, \{w_4, w_5\})\}$.

Suppose $M, u \models p$ and consider the specification $\sigma = [p \mapsto a]^1$. Since p holds at the root, player 1 is restricted to make the choice "a" at u. Hence the relation in this case would be $R_{(g,\sigma)} = \{(u, \{w_1, w_2, w_3\}), (v_1, \{w_1, w_2, w_3\}), (v_2, \{w_4, w_5\})\}$. □

Example 4.2.4 For a game g and a specification σ of player i, the formula $\langle\langle (g, \sigma)\rangle\rangle^{\forall}\alpha$ asserts that the game g is enabled and player i has a strategy in g conforming to σ to ensure α.

The logic is also powerful enough to assert the non-existence of strategies for a player with respect to ensuring an outcome α. For a game g, consider the formula

- $\alpha' = \langle\langle (g, null^i)\rangle\rangle^{\forall} True \wedge \neg\langle\langle (g, null^i)\rangle\rangle^{\forall}\alpha$.

The first conjunct $\langle\langle (g, null^i)\rangle\rangle^{\forall} True$ asserts the fact that game g is enabled. Given that g is enabled, the only way $\neg\langle\langle (g, null^i)\rangle\rangle^{\forall}\alpha$ can be true is if player i does not

have a strategy conforming to $null^i$ which ensures α. Recall that any strategy of player i conforms to $null^i$. Thus α' holds at a state u iff player i does not have a strategy at u that ensures the objective α. □

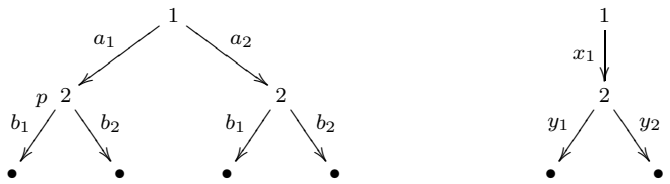


Figure 4.4: Composition of games

Example 4.2.5 To illustrate compositional reasoning in the logic, consider the games g_1 and g_2 given in Fig. 4.4. Let u be a state of the model where g_1 is enabled. Let g denote the game $g_1; g_2$, i.e. the game obtained by pasting g_2 at each of the leaf nodes of g_1. We use the following notations:

- w^{a_1}: denotes the state reached after action a_1.
- w^{a_1,b_1}: the state reached on following actions a_1 and b_1.
- $w^{a_1,b_1}_{x_1,y_1}$: the state reached on the sequence of actions $a_1b_1x_1y_1$ where a_1b_1 are actions in game g_1 and x_1y_1 actions in game g_2.

Let win_1, win_2 and p be propositions whose valuations are given by $V(win_2) = \{w^{a_1,b_1}, w^{a_2,b_2}\}$, $V(win_1) = \{w^{a_1,b_1}_{x_1,y_1}, w^{a_2,b_2}_{x_1,y_2}\}$ and $V(p) = \{w^{a_1}\}$. Consider the following specifications:

- $\pi = [p \mapsto b_1]^2 \cdot [\neg p \mapsto b_2]^2$.
- $\sigma = [True \mapsto x_1]^1$.

It is easy to see that $\langle (g_1, \pi) \rangle win_2$ holds at u. Player 1 does not have a strategy in the composite game g to ensure win_1. However, in the composite pair $\xi = (g_1, \pi); (g_2, \sigma)$, it is easy to see that $\langle \xi \rangle win_1$ holds. Assuming that in the game g_1 player 2 plays according to π then in g_2 by using a strategy which conforms to σ player 1 can ensure win_1. □

In some sense the above example says that reasoning in the game g is different from reasoning in g_1 followed by g_2. In the latter, the additional structural information is available which can be used for strategizing. For simple game structures it is quite obvious that such reasoning can be done with a past modality. It is iteration which provides the actual expressive power. In the presence of iteration, the analysis asserts the fact that players can take into account the structure of the game and the opponent's strategy. In particular while strategizing, a player can make use of the fact that the opponent is using a bounded memory strategy and that with the type of strategy that is being used the opponent can be forced into a particular region of the game graph.

The above mentioned reasoning can also be thought of as players trying to attain certain local goals. If player 2 plays to achieve the local goal win_2 then player 1 can use this information and respond with a strategy in g_2 to achieve the objective win_1. Players can then try to achieve their global objective by performing appropriate composition of the local objectives.

4.2.3 Encoding PDL

In this section we show that the propositional dynamic logic (PDL) (introduced in Section 2.4) can be encoded into the dynamic logic on compositional games. We look at the "test free" version of PDL. In Section 4.2.6 we show how the test operator can be added to the dynamic logic on compositional games. Thus it follows that full PDL can also be encoded in the logic.

Translation

In order to translate PDL formulas into formulas in our logic, we make use of the following games: $g_a^i = ((i,x),a,(j,y))$ and $g_a^{\overline{i}} = ((\overline{i},x),a,(j,y))$. For a program $\gamma \in \mathbb{P}$, we translate γ into a composite game-strategy pair $\xi \in \Gamma$ inductively as follows:

$$\begin{aligned} \|a\| &= (g_a^i, [\mathit{True} \mapsto a]^i) \cup (g_a^{\overline{i}}, [\mathit{True} \mapsto a]^{\overline{i}}) \\ \|\gamma_1;\gamma_2\| &= \|\gamma_1\};\|\gamma_2\| \\ \|\gamma_1 \cup \gamma_2\| &= \|\gamma_1\| \cup \|\gamma_2\| \\ \|\gamma_1^*\| &= \|\gamma_1\|^* \end{aligned}$$

Proposition 4.2.6 *For all M and $\gamma \in \mathbb{P}$ if $(u,X) \in R_{\|\gamma\|}$ then $|X| = 1$.*

Proof: The proposition can be easily verified, the interesting case is when we have an atomic program, i.e. $\gamma = a \in \Sigma$. In this case we have $\|a\| = (g_a^i,\ [True \mapsto a]^i) \cup (g_a^{\overline{i}}, [True \mapsto a]^{\overline{i}})$. Suppose $(u, X) \in R_{\|\gamma\|}$, assume without loss of generality that $(u, X) \in R_{(g_a^i, [True \mapsto a]^i)}$. From semantics we have that the game g_a^i is enabled at u and there exists a strategy for player i which conforms to $[True \mapsto a]^i$. Since g_a^i is enabled at u we have that $\lambda(u) = i$ and player i's strategy needs to choose a unique edge in the tree $T_u \upharpoonright g_a^i$, it follows that $|X| = 1$. □

For a pair $(u, \{w\})$, let $map(u, \{w\}) = (u, w)$. We extend map to sets of such pairs as $map(Y) = \{map(y) \mid y \in Y\}$. For $\gamma \in \mathbb{P}$, due to proposition 4.2.6 it follows that $map(R_{\|\gamma\|})$ is well defined.

Lemma 4.2.7 *For all M, for all $\gamma \in \mathbb{P}$, $R_\gamma^{PDL} = map(R_{\|\gamma\|})$.*

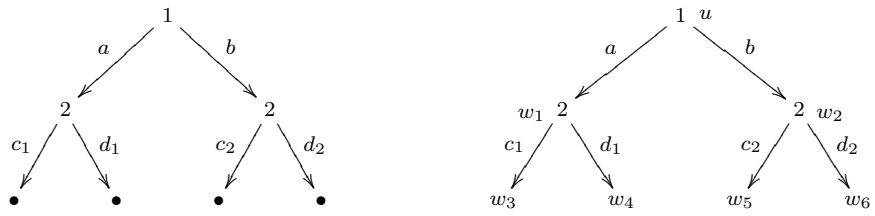
Proof: The proof is by induction on the structure of γ.
$\gamma = a$. Suppose $(u, w) \in R_\gamma^{PDL}$. This implies that $u \xrightarrow{a} w$ in the Kripke structure M. Without loss of generality assume that $turn(u) = i$, then we have g_a^i is enabled at u. Since $[True \mapsto a]^i$ is a player i specification and $u \xrightarrow{a} w$ we get $(u, \{w\}) \in R_{\|\gamma\|}$.

Conversely, suppose $(u, \{w\}) \in R_{\|\gamma\|}$. Assume that $turn(u) = i$, from semantics we have $(u, \{w\}) \in R_{(g_a^i, [True \mapsto a]^i)}$. This implies that g_a^i is enabled at u and thus $u \xrightarrow{a} w$. Therefore we get $(u, w) \in R_\gamma^{PDL}$.
$\gamma = \gamma_1 \cup \gamma_2$. The claim easily follows from application of induction hypothesis.
$\gamma = \gamma_1; \gamma_2$. Suppose $(u, w) \in R_\gamma^{PDL}$, from semantics we have that there exists a v such that $(u, v) \in R_{\gamma_1}^{PDL}$ and $(v.w) \in R_{\gamma_2}^{PDL}$. By induction hypothesis we have $(u, \{v\}) \in R_{\|\gamma_1\|}$ and $(v, \{w\}) \in R_{\|\gamma_2\|}$. From semantics we get $(u, \{w\}) \in R_{\|\gamma\|}$. Using a similar argument we can show that if $(u, \{w\}) \in R_{\|\gamma\|}$ then $(u, w) \in R_\gamma^{PDL}$.
$\gamma = \gamma_1^*$. Suppose $(u, w) \in R_{\gamma_1^*}^{PDL}$, from semantics there exists a k such that $(u, w) \in (R_{\gamma_1}^{PDL})^k$. By a second induction on k we can show that $(u, \{w\}) \in (R_{\|\gamma_1\|})^k$, which implies $(u, \{w\}) \in (R_{\|\gamma_1^*\|})$ as well. A similar argument also shows that if $(u, \{w\}) \in R_{\|\gamma_1^*\|}$ then $(u, w) \in R_{\gamma_1^*}^{PDL}$. □

The translation function $\|\cdot\|$ can be extended to formulas of PDL in the obvious manner where:

- $\|\langle\gamma\rangle\alpha\| = \langle\|\gamma\|\rangle^\forall \|\alpha\|$.

Theorem 4.2.8 then follows from Lemma 4.2.7 by an inductive argument.



(a): Game g (b): Model M

Figure 4.5: Extensive form game tree and model

Theorem 4.2.8 *For all M, for states $w \in W$ and for all $\alpha \in$ **PDL**, $M, w \models \alpha$ iff $M, w \models \|\alpha\|$.*

4.2.4 Axiom system

We now present an axiomatization of the valid formulas of the logic. For a set $A = \{a_1, \ldots, a_k\} \subseteq \Sigma$, we will use the notation $\Re(i, x, A)$ to denote the game $((i, x), a_1, t_{a_1} + \cdots + (i, x), a_k, t_{a_k})$.

We also make use of the following abbreviations:

- Let $g_a^i = ((i, x), a, (j, y))$ and $g_a^{\overline{i}} = ((\overline{i}, x), a, (j, y))$,
  - $\langle a \rangle \alpha \equiv \langle (g_a^i, [True \mapsto a]^i) \cup (g_a^{\overline{i}}, [True \mapsto a]^{\overline{i}} \rangle^{\forall} \alpha$.

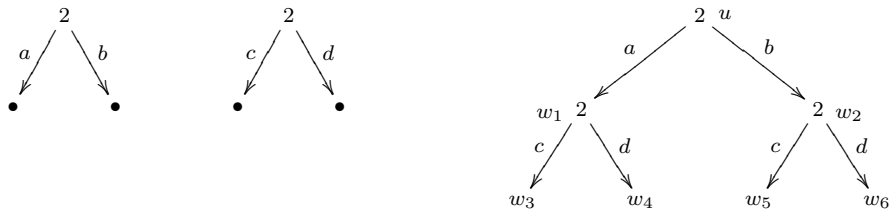
From Theorem 4.2.8, it follows that this results in the standard interpretation for $\langle a \rangle \alpha$, i.e. $\langle a \rangle \alpha$ holds at a state u iff there is a state w such that $u \xrightarrow{a} w$ and α holds at w.

For game g, we use the formula $g^{\surd}$ to denote that the game structure g is enabled. This is defined as:

- For $g = (i, x)$, let $g^{\surd} = True$.
- For $g = \Re(i, x, A)$, let
  - $g^{\surd} = \mathbf{turn}_i \wedge (\bigwedge_{j=1,\ldots,k} (\langle a_j \rangle True \wedge [a_j] t_{a_j}^{\surd}))$.

Proposition 4.2.9 *For $\alpha_1, \alpha_2 \in \Phi$, the following holds:*

1. *for $\xi \in \Gamma$, the formula $\langle \xi \rangle^{\forall} (\alpha_1 \vee \alpha_2) \supset \langle \xi \rangle^{\forall} \alpha_1 \vee \langle \xi \rangle^{\forall} \alpha_2$ is not valid.*
2. *$\langle a \rangle (\alpha_1 \vee \alpha_2) \equiv \langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2$ is valid.*



(a): g_1 (b): g_2 (b): Model M

Figure 4.6: Extensive form game tree and model

Proof: (1): Let g be the game given in Figure 4.5(a) and the model M be as shown in Figure 4.5(b). Let propositions $p \in V(u)$, $q_1 \in V(w_3)$ and $q_2 \in V(w_4)$. Let $\alpha_1 = q_1$, $\alpha_2 = q_2$ and $\sigma = [p \mapsto a]^1$. We have $M, u \models \langle g, \sigma \rangle^{\forall}(q_1 \vee q_2)$ since the strategy of player 1 which chooses the move a at u ensures $q_1 \vee q_2$. However since $q_1 \notin V(w_4)$ and $q_2 \notin V(w_3)$, we obtain $M, u \not\models \langle g, \sigma \rangle^{\forall} q_1 \vee \langle g, \sigma \rangle^{\forall} q_2$.

(2): We first show that $\langle a \rangle(\alpha_1 \vee \alpha_2) \supset \langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2$ is valid. Suppose this is not true; there exists a model M and a state u such that $M, u \models \langle a \rangle(\alpha_1 \vee \alpha_2)$ and $M, u \not\models \langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2$. This implies that $M, u \not\models \langle a \rangle \alpha_1$ and $M, u \not\models \langle a \rangle \alpha_2$. Thus for all states w such that $u \xrightarrow{a} w$ we have $M, w \not\models \alpha_1$ and $M, w \not\models \alpha_2$. From this we can infer that $M, u \not\models \langle a \rangle(\alpha_1 \vee \alpha_2)$ contradicting the assumption.

The fact that $\langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2 \supset \langle a \rangle(\alpha_1 \vee \alpha_2)$ is valid follows from a similar argument. □

Proposition 4.2.10 *For $\xi_1, \xi_2 \in \Gamma$, consider the usual relation composition semantics for $R_{\xi_1;\xi_2}$, i.e. $R_{\xi_1;\xi_2} = \{(u, X) \mid \exists Y$ such that $(u, Y) \in R_{\xi_1}$ and for all $v \in Y$, $(v, X) \in R_{\xi_2}\}$. Under this interpretation, the formula $\langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha \supset \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$ is not valid.*

Proof: Let g_1 and g_2 be the games given in Figure 4.6(a) and (b) respectively and let the model M be given in Figure 4.6(c). Let the proposition p hold at states w_3, w_4, w_5 and w_6 and $\alpha = p$. Let $\xi_1 = (g_1, null^1)$ and $\xi_2 = (g_2, null^1)$, then we have $M, u \models \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} p$. However, under the interpretation given in the statement of the proposition, for $X = \{w_3, w_4, w_5, w_6\}$ we have $(u, X) \notin R_{\xi_1;\xi_2}$. Thus $M, u \not\models \langle \xi_1; \xi_2 \rangle^{\forall} p$. □

Proposition 4.2.11 *The formula* $\langle \xi_1; \xi_2 \rangle^{\forall} \alpha \equiv \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$ *is valid.*

Proof: Suppose $\langle \xi_1; \xi_2 \rangle^{\forall} \alpha \supset \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$ is not valid. Then there exists M and u such that $M, u \models \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$ and $M, u \not\models \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$. Since $M, u \models \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$, from semantics we have there exists $(u, X) \in R_{\xi_1;\xi_2}$ such that $\forall w \in X$, $M, w \models \alpha$. From definition of R, $\exists Y = \{v_1, \ldots, v_k\}$ such that $(u, Y) \in R_{\xi_1}$ and $\forall v_j \in Y$ there exists $X_j \subseteq X$ such that $(v_j, X_j) \in R_{\xi_2}$ and $\bigcup_{j=1,\ldots,k} X_j = X$. Therefore we get $\forall v_k \in Y$, $M, v_k \models \langle \xi_2 \rangle^{\forall} \alpha$ and hence from semantics, $M, u \models \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$. This gives the required contradiction.

Suppose $\langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha \supset \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$ is not valid. Then there exists M and u such that $M, u \models \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$ and $M, u \not\models \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$. We have $M, u \models \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$ iff there exists $(u, Y) \in R_{\xi_1}$ such that $\forall v_k \in Y$, $M, v_k \models \langle \xi_2 \rangle^{\forall} \alpha$. $M, v_k \models \langle \xi_2 \rangle^{\forall} \alpha$ iff there exists $(v_k, X_k) \in R_{\xi_2}$ such that $\forall w_k \in X_k$, $M, w_k \models \alpha$. Let $X = \bigcup_k X_k$, from definition of R we get $(u, X) \in R_{\xi_1;\xi_2}$. Hence from semantics $M, u \models \langle \xi_1; \xi_2 \rangle^{\forall} \alpha$. □

The axiom schemes

(A1) Propositional axioms:

- (a) All the substitutional instances of tautologies of PC.
- (b) $\mathbf{turn}_i \equiv \neg \mathbf{turn}_{\overline{i}}$.

(A2) Axiom for single edge games:

- (a) $\langle a \rangle (\alpha_1 \vee \alpha_2) \equiv \langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2$.
- (b) $\langle a \rangle \mathbf{turn}_i \supset [a] \mathbf{turn}_i$.

(A3) Dynamic logic axioms:

- (a) $\langle \xi_1 \cup \xi_2 \rangle^{\forall} \alpha \equiv \langle \xi_1 \rangle^{\forall} \alpha \vee \langle \xi_2 \rangle^{\forall} \alpha$.
- (b) $\langle \xi_1; \xi_2 \rangle^{\forall} \alpha \equiv \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$.
- (c) $\langle \xi^* \rangle^{\forall} \alpha \equiv \alpha \vee \langle \xi \rangle^{\forall} \langle \xi^* \rangle^{\forall} \alpha$.

(A4) $\langle g, \sigma \rangle^{\forall} \alpha \equiv g^{\checkmark} \wedge \mathit{push}(g, \sigma, \alpha)$.

Inference rules

$$(MP)\ \frac{\alpha, \quad \alpha \supset \beta}{\beta} \qquad (NG)\ \frac{\alpha}{[a]\alpha}$$

$$(IND)\ \frac{\langle \xi \rangle^{\forall} \alpha \supset \alpha}{\langle \xi^* \rangle^{\forall} \alpha \supset \alpha}$$

Definition of *push*:

Axiom (A3) provides the reduction axioms for composite game strategy pairs. However, the atomic case (g, σ) still encodes the structure of the game tree g and the strategy specification σ explicitly. Thus we need to provide the reduction axioms for atomic game strategy pairs as well. This is done in axiom (A4). The intuitive meaning of the construct $push(g, \sigma, \alpha)$ is to say that player i has a strategy conforming to σ such that α holds at all the frontier nodes. As expected this needs to be defined inductively on the structure of σ and the game tree g. The core idea in the definition is the following observation: For a strategy specification σ and a game g,

- if the root of the game tree g is an i node then by definition of a strategy, player i needs to choose an edge 'a' which conforms to σ and the resulting subtree g_a needs to inductively satisfy $\langle g_a, \sigma \rangle^{\forall} \alpha$.
- if the root is an $\overline{i}$ node then all outgoing edges need to be considered (by definition of a strategy) and for each edge 'a', the requirement $\langle g_a, \sigma \rangle^{\forall} \alpha$ needs to be pushed to the resulting subtree.

The formal definition is given below. For $i \in N$ let $\sigma, \sigma_1, \sigma_2 \in Strat^i(P^i)$ and let $\pi \in Strat^{\overline{i}}(P^{\overline{i}})$. Let $g \in \mathbb{G}(Nodes)$ and $\alpha \in \Phi$. For $push(g, \sigma, \alpha)$ we have various cases depending on the structure of g.

The case when g is an atomic game, i.e. $g = (i, x)$, for all $i \in N$ and $\sigma \in Strat^i(P^i)$ we have,

(C1) $push(g, \sigma, \alpha) = \alpha$.

Suppose $g = \Re(i, x, A)$ for $A = \{a_1, \ldots, a_k\}$, i.e. g is a tree with root being an i node. For each $a_m \in A$ let $g_{a_m} = ((i, x), a_m, (j_m, y_m))$, where (j_m, y_m) is the root of t_{a_m}.

(C2) $push(g, [\psi \mapsto a]^i, \alpha) =$
$(\psi \supset \langle a \rangle \langle t_a, [\psi \mapsto a]^i \rangle^{\forall} \alpha) \wedge (\neg\psi \supset (\bigvee_{a_m \in A} \langle a_m \rangle \langle t_{a_m}, [\psi \mapsto a]^i \rangle^{\forall} \alpha))$.

The root of g is a player i node and case (C2) is a player i specification. This says that if ψ holds at the root, then there is a choice a that player i can make such that for the subtree t_a, $\langle t_a, [\psi \mapsto a]^i \rangle^{\forall} \alpha$ holds. If ψ does not hold at the root then there is some choice a_m for player i such that the subtree t_{a_m} satisfies $\langle t_{a_m}, [\psi \mapsto a]^i \rangle^{\forall} \alpha$.

(C3) $push(g, \sigma_1 \cdot \sigma_2, \alpha) = \bigvee_{a_m \in A} (\langle g_{a_m}, \sigma_1 \rangle^{\forall} \langle t_{a_m}, \sigma_1 \cdot \sigma_2 \rangle^{\forall} \alpha \wedge \langle g_{a_m}, \sigma_2 \rangle^{\forall} \langle t_{a_m}, \sigma_1 \cdot \sigma_2 \rangle^{\forall} \alpha)$.

For (C3) the important point to note is the fact if an edge $u \xrightarrow{a} w$ satisfies a specification σ then all w' with $u \xrightarrow{a} w'$ satisfies σ. This is because satisfaction of σ depends only on u and the action a, it does not depend on the target node.

(C4) $push(g, \sigma_1 + \sigma_2, \alpha) = \bigvee_{a_m \in A} (\langle g_{a_m}, \sigma_1 \rangle^{\forall} \langle t_{a_m}, \sigma_1 + \sigma_2 \rangle^{\forall} \alpha \vee \langle g_{a_m}, \sigma_2 \rangle^{\forall} \langle t_{a_m}, \sigma_1 + \sigma_2 \rangle^{\forall} \alpha)$.

(C5) $push(g, \pi \Rightarrow \sigma, \alpha) = \bigvee_{a_m \in A} (\langle g_{a_m}, \sigma \rangle^{\forall} \langle t_{a_m}, \pi \Rightarrow \sigma \rangle^{\forall} \alpha)$.

Suppose $g = \Re(\overline{i}, x, A)$ for $A = \{a_1, \ldots, a_k\}$, i.e. g is a tree with root being an $\overline{i}$ node. For each $a_m \in A$ let $g_{a_m} = ((i, x), a_m, (j_m, y_m))$, where (j_m, y_m) is the root of t_{a_m}. For $\sigma = [\psi \mapsto a]^i, \sigma_1 + \sigma_2, \sigma_1 \cdot \sigma_2$,

(C6) $push(g, \sigma, \alpha) = \bigwedge_{a_m \in A} [a_m] \langle t_{a_m}, \sigma \rangle^{\forall} \alpha$.

(C6) says that when the root node of g is an $\overline{i}$ node and for a player i specification which is not of the form $\pi \Rightarrow \sigma$, if at all enabled edges a_m, the subtree t_{a_m} satisfies $\langle t_{a_m}, \sigma \rangle^{\forall} \alpha$ then $\langle g, \sigma \rangle^{\forall} \alpha$ holds.

(C7) $push(g, \pi \Rightarrow \sigma, \alpha) =$
$$\bigwedge_{a_m \in A} ((\langle g_{a_m}, \pi \rangle^{\forall} True \supset [a_m] \langle t_{a_m}, \pi \Rightarrow \sigma \rangle^{\forall} \alpha) \wedge (\neg \langle g_{a_m}, \pi \rangle^{\forall} True \supset [a_m] \langle t_{a_m}, null^i \rangle^{\forall} \alpha)).$$

The interesting case is when the root of g is an $\overline{i}$ node and when the specification is of the form $\pi \Rightarrow \sigma$, this is specified in (C7). For a strategy μ of player i to satisfy $\pi \Rightarrow \sigma$ on g, it should make sure of the following:

- for each choice $a_m \in A$, if the choice conforms with π then the strategy on t_{a_m} should satisfy σ.
- for each choice $a_m \in A$, which does not conform with π, player i is allowed to employ any strategy on the game t_{a_m}.

4.2.5 Completeness

To show completeness, we prove that every consistent formula is satisfiable. Let α_0 be a consistent formula, and $CL(\alpha_0)$ denote the subformula closure of α_0. In addition to the usual downward closure, we also require that $\langle g, \sigma \rangle^{\forall} \alpha \in CL(\alpha_0)$ implies that $g^{\surd}, push(g, \sigma, \alpha) \in CL(\alpha_0)$. Let $AT(\alpha_0)$ be the set of all maximal consistent subsets of $CL(\alpha_0)$, referred to as atoms. Each $\mathbf{t} \in AT(\alpha_0)$ is a finite set of formulas, we

denote the conjunction of all formulas in $\mathfrak{t}$ by $\widehat{\mathfrak{t}}$. For a nonempty subset $X \subseteq AT$, we denote by $\widetilde{X}$ the disjunction of all $\widehat{\mathfrak{t}}, \mathfrak{t} \in X$. Define a transition relation on $AT(\alpha_0)$ as follows: $\mathfrak{t} \xrightarrow{a}_{AT} \mathfrak{t}'$ iff $\widehat{\mathfrak{t}} \wedge \langle a \rangle \widehat{\mathfrak{t}'}$ is consistent. The model $M = (W, \rightarrow, \lambda, V)$ where $W = AT(\alpha_0)$ and $\rightarrow = \rightarrow_{AT}$. Note that each $w \in W$ is an atom and thus we use the notation $\widehat{w}$ to denote the conjunction of all formulas in w. The valuation V is defined as $V(w) = \{p \in P \mid p \in w\}$ and $\lambda(w) = i$ iff $\mathbf{turn}_i \in w$. Once the Kripke structure is defined, the game theoretic semantics given earlier defines the relation $R_{(g,\sigma)}$ on $W \times 2^W$ for $g \in \mathbb{G}(Nodes)$ and a strategy specification σ.

The following lemma can be shown using standard modal logic techniques.

Lemma 4.2.12 *For all $u \in W$, for all $\alpha \in CL(\alpha_0)$, for all $a \in \Sigma$, if for all v such that $u \xrightarrow{a} v$ we have $\widehat{v} \wedge \alpha$ is consistent then $\widehat{u} \wedge [a]\alpha$ is consistent.*

Lemma 4.2.13 *For all $g \in \mathbb{G}(Nodes)$, for all $i \in N$ and $\sigma \in Strat^i(P^i)$, for all $X \subseteq W$ and for all $u \in W$ the following holds:*

($\mathcal{L}1$) if $(u, X) \in R_{(g,\sigma)}$ then $\widehat{u} \wedge \langle g, \sigma \rangle^{\forall} \widetilde{X}$ is consistent.

($\mathcal{L}2$) if $\widehat{u} \wedge \langle g, \sigma \rangle^{\forall} \widetilde{X}$ is consistent then there exists $X' \subseteq X$ such that $(u, X') \in R_{(g,\sigma)}$.

Proof: By induction on the structure of σ.
$\sigma = [\psi \mapsto a]^i$:

($\mathcal{L}1$) Suppose $(u, X) \in R_{(g,[\psi \mapsto a]^i)}$, we need to show that $\widehat{u} \wedge \langle g, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. We do a second induction on the structure of g. The base case is when $g = (i, x)$, and the claim follows easily from axiom (A4) case (C1).

Let $g = \Re(i, x, A)$ where $A = \{a_1, \ldots, a_k\}$. Suppose $(u, X) \in R_{(g,[\psi \mapsto a]^i)}$, since $enabled(g, u)$ holds we have there exist sets $Y_1, \ldots, Y_k$ such that for all $j : 1 \leq j \leq k$, for all $w_j \in Y_j$ we have $u \xrightarrow{a_j} w_j$. Since u is an i node, any strategy of i will pick a unique edge at u. We have the following two cases:

- $M, u \models \psi$: From semantics, the strategy should choose a w_j such that $a_j = a$, $u \xrightarrow{a_j} w_j$ and $(w_j, X) \in R_{(t_j,[\psi \mapsto a]^i)}$. By the secondary induction hypothesis, we have $\widehat{w_j} \wedge \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. Hence $\widehat{u} \wedge \langle a \rangle \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent.

- $M, u \not\models \psi$: The strategy can choose any w_j such that $u \xrightarrow{a_j} w_j$ and $(w_j, X) \in R_{(t_j,[\psi \mapsto a]^i)}$. By the secondary induction hypothesis, $\widehat{w_j} \wedge \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. Hence $\widehat{u} \wedge \langle a_j \rangle \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent.

From axiom (A4) case (C2) we get $\widehat{u} \wedge \langle g, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent.

Let $g = \Re(\overline{i}, x, A)$ where $A = \{a_1, \ldots, a_k\}$. Suppose $(u, X) \in R_{(g,[\psi \mapsto a]^i)}$, since $enabled(g, u)$ holds, we have there exist $Y_1, \ldots, Y_k$ such that for all $j : 1 \leq j \leq k$, for all $w_j \in Y_j$, $u \xrightarrow{a_j} w_j$. Since u is an $\overline{i}$ node, any strategy μ of i conforming to $[\psi \mapsto a]^i$ will have all the branches at u (by definition of strategy). Therefore we get for all w_j with $u \xrightarrow{a_j} w_j$, there exists $X_j \subseteq X$ such that $(w_j, X_j) \in R_{(t_j,\pi)}$ and $X = \bigcup_{j=1,\ldots,k} X_j$. By secondary induction hypothesis and the fact that $X_j \subseteq X$, we have $\widehat{w}_j \wedge \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. Hence from axiom (A4) case (C6), we conclude that $\widehat{u} \wedge \langle g, \sigma \rangle^{\forall} \widetilde{X}$ is consistent.

($\mathcal{L}2$) Suppose $\widehat{u} \wedge \langle g, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent, we need to show that there exists $X' \subseteq X$ such that $(u, X') \in R_{(g,[\psi \mapsto a]^i)}$. We do a second induction on the structure of g. The base case is when $g = (i, x)$, and the claim follows easily from axiom (A4) case (C1).
Let $g = \Re(i, x, A)$ where $A = \{a_1, \ldots, a_k\}$. From axiom (A4) it follows that there exist sets $Y_1, \ldots, Y_k$ such that for all $j : 1 \leq j \leq k$, for all $w_j \in Y_j$ we have $u \xrightarrow{a_j} w_j$ and hence $enabled(g, u)$ holds. Let $X = \{v_1, \ldots, v_m\}$. We have the following two cases:

- if $M, u \models \psi$: then from case (C2), $\widehat{u} \wedge \langle a \rangle \langle t_a, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. Hence we get there exists w_a such that $u \xrightarrow{a} w_a$ and $\widehat{w}_a \wedge \langle t_a, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. By the secondary induction hypothesis there exists $X' \subseteq X$ such that $(w_a, X') \in R_{(t_a,[\psi \mapsto a]^i)}$ and by definition of R we have $(u, X') \in R_{(g,[\psi \mapsto a]^i)}$.
- if $M, u \not\models \psi$: then from case (C2), $\widehat{u} \wedge \bigvee_{a_j \in A} \langle a_j \rangle \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$. Therefore there exists w_j such that $u \xrightarrow{a_j} w_j$ and $\widehat{w}_j \wedge \langle t_j, [\psi \mapsto a]^i \rangle^{\forall} \widetilde{X}$ is consistent. By the secondary induction hypothesis there exists $X' \subseteq X$ such that $(w_j, X') \in R_{(t_j,[\psi \mapsto a]^i)}$ and therefore we have $(u, X') \in R_{(g,[\psi \mapsto a]^i)}$.

For the case when $g = \Re(\overline{i}, x, A)$ where $A = \{a_1, \ldots, a_k\}$ the claim can be shown using Lemma 4.2.12 and axiom (A4) case (C6).
$\sigma = \sigma_1 + \sigma_2$:

Again we do a second induction on the structure of g. The base case when $g = (i, x)$ follows easily from axiom (A4) case (C1).
Let $g = \Re(i, x, A)$ where $A = \{a_1, \ldots, a_k\}$. Suppose $(u, X) \in R_{(g,\sigma_1+\sigma_2)}$, since $enabled(g, u)$ holds we have there exist sets $Y_1, \ldots, Y_k$ such that for all $j : 1 \leq j \leq k$,

for all $w_j \in Y_j$ we have $u \xrightarrow{a_j} w_j$. Since u is an i node, from semantics we have there exists a w_j such that $(u, \{w_j\}) \in R_{(g_{a_j}, \sigma_1)}$ or $(u, \{w_j\}) \in R_{(g_{a_j}, \sigma_2)}$ and $(w_j, X) \in R_{(t_{a_j}, \sigma_1 + \sigma_2)}$. By the main induction hypothesis, we get $\widehat{u} \wedge \langle g_j, \sigma_1 \rangle^{\forall} \widehat{w_j}$ is consistent or $\widehat{u} \wedge \langle g_j, \sigma_2 \rangle^{\forall} \widehat{w_j}$ is consistent. By secondary induction hypothesis we get $\widehat{w_j} \wedge \langle t_j, \sigma_1 + \sigma_2 \rangle^{\forall} \widetilde{X}$ is consistent. From axiom (A4) case (C4) we get $\widehat{u} \wedge \langle g, \sigma_1 + \sigma_2 \rangle^{\forall} \widetilde{X}$ is consistent.

The case when $g = \Re(\overline{\imath}, x, A)$ where $A = \{a_1, \ldots, a_k\}$ follows from a similar argument making use of axiom (A4) case (C6). ($\mathcal{L}1$) can also be shown by making use of axioms (A4) cases (C4) and (C6) and application of the induction hypothesis. For $\sigma = \sigma_1 \cdot \sigma_2$, the result follows from axiom (A4) cases (C3) and (C6) using similar arguments.

$\sigma = \pi \Rightarrow \sigma_1$:

The cases when g is of the form (i, x) and $g = \Re(i, x, A)$ for $A = \{a_1, \ldots, a_k\}$ follows from axiom (A4) cases (C1) and (C5) respectively. The interesting case is when $g = \Re(\overline{\imath}, x, A)$ for $A = \{a_1, \ldots, a_k\}$.

Suppose $(u, X) \in R_{(g, \pi \Rightarrow \sigma)}$, since $enabled(g, u)$ holds, it is easy to show that $\widehat{u} \wedge g^{\checkmark}$ is consistent. We also get that there exist sets $Y_1, \ldots, Y_k$ such that for all $j : 1 \leq j \leq k$, for all $w_j^r \in Y_j$ we have $u \xrightarrow{a_j} w_j^r$. Since u is an $\overline{\imath}$ node, from semantics we get $(u, X) \in R_{(g, \pi \Rightarrow \sigma)}$ iff for all $a_j \in A$ for all $w_j^r \in Y_j$, there exists X_j^r such that one of the following conditions hold.

P1: if $(u, w_j^r) \in R_{(g_j, \pi)}$ then $(w_j^r, X_j^r) \in R_{(t_j^r, \sigma \Rightarrow \pi)}$.

P1: if $(u, w_j^r) \notin R_{(g_j, \pi)}$ then $(w_j^r, X_j^r) \in R_{(t_j^r, null^i)}$.

We also have that $X = \bigcup_{j=1,\ldots,k} \bigcup_{r=1,\ldots,|Y_j|} X_j^r$.

Note that from the semantics it follows that for any $a_j \in A$ and $w_j \in Y_j$, if $(u, w_j) \in R_{(g_j, \pi)}$ then for all $w_j^r \in Y_j$, $(u, w_j^r) \in R_{(g_j, \pi)}$. Thus if **P1** holds for a_j then by main induction hypothesis we get $\widehat{u} \wedge \langle (g_j, \pi) \rangle^{\forall} \widehat{w_j^r}$ is consistent for all $w_j^r \in Y^j$ and thus $\widehat{u} \wedge \langle g_j, \pi \rangle^{\forall} True$ is consistent. By secondary induction hypothesis and the fact that $X_j^r \subseteq X$ we have $\widehat{w_j^r} \wedge \langle (t_j, \sigma \Rightarrow \pi) \rangle^{\forall} \widetilde{X}$ is consistent for all $w_j^r \in Y_j$. From Lemma 4.2.12 we have $\widehat{u} \wedge [a_j] \langle t_j, \sigma \Rightarrow \pi \rangle^{\forall} \widetilde{X}$ is consistent and therefore $\langle g_j, \pi \rangle^{\forall} True \supset [a_j] \langle t_j, \sigma \Rightarrow \pi \rangle^{\forall} \widetilde{X}$ is consistent.

If **P2** holds for a_j then by main induction hypothesis of ($\mathcal{L}2$) we can deduce that $\widehat{u} \wedge \neg \langle g_j, \pi \rangle^{\forall} \widehat{w_j^r}$ is consistent. By secondary induction hypothesis we have $\widehat{w_j^r} \wedge \langle t_{a_j}, null^i \rangle^{\forall} \widetilde{X}$ is consistent.

From axiom (A4) case (C7) we get that $\widehat{u} \wedge \langle g, \pi \Rightarrow \sigma_1 \rangle^{\forall} \widetilde{X}$ is consistent.

($\mathcal{L}1$) can also be shown by making use of axioms (A4) cases (C5) and (C7). $\square$

Lemma 4.2.14 *For all $\xi \in \Gamma$, for all $X \subseteq W$ and $u \in W$, if $\widehat{u} \wedge \langle \xi \rangle^{\forall} \widetilde{X}$ is consistent then there exists $X' \subseteq X$ such that $(u, X') \in R_\xi$.*

Proof: By induction on the structure of ξ.

- $\xi = (g, \sigma)$: Suppose $\widehat{u} \wedge \langle g, \sigma \rangle^{\forall} \widetilde{X}$ is consistent. From Lemma 4.2.13 item 2, it follows that there exists $X' \subseteq X$ such that $(u, X') \in R_\xi$.

- $\xi = \xi_1 \cup \xi_2$: By axiom (A3a) we get $\widehat{u} \wedge \langle \xi_1 \rangle^{\forall} \widetilde{X}$ is consistent or $\widehat{u} \wedge \langle \xi_2 \rangle^{\forall} \widetilde{X}$ is consistent. By induction hypothesis there exists $X_1 \subseteq X$ such that $(u, X_1) \in R_{\xi_1}$ or there exists $X_2 \subseteq X$ such that $(u, X_2) \in R_{\xi_2}$. Hence we have $(u, X_1) \in R_{\xi_1 \cup \xi_2}$ or $(u, X_2) \in R_{\xi_1 \cup \xi_2}$.

- $\xi = \xi_1; \xi_2$: By axiom (A3b), $\widehat{u} \wedge \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \widetilde{X}$ is consistent. Hence $\widehat{u} \wedge \langle \xi_1 \rangle^{\forall} (\bigvee (\widehat{w} \wedge \langle \xi_2 \rangle^{\forall} \widetilde{X}))$ is consistent, where the join is taken over all $w \in Y = \{w \mid w \wedge \langle \xi_2 \rangle^{\forall} \widetilde{X} \text{ is consistent}\}$. So $\widehat{u} \wedge \langle \xi_1 \rangle^{\forall} \widetilde{Y}$ is consistent. By induction hypothesis on ξ_1, there exists $Y' \subseteq Y$ such that $(u, Y') \in R_{\xi_1}$. We also have that for all $w \in Y$, $\widehat{w} \wedge \langle \xi_2 \rangle^{\forall} \widetilde{X}$ is consistent. Therefore we get for all $w_j \in Y' = \{w_1, \ldots, w_k\}$, $\widehat{w}_j \wedge \langle \xi_2 \rangle^{\forall} \widetilde{X}$ is consistent. By induction hypothesis on ξ_2, there exists $X_j \subseteq X$ such that $(w_j, X_j) \in R_{\xi_2}$. Let $X' = \bigcup_{j=1,\ldots,k} X_j \subseteq X$, we get $(u, X') \in R_{\xi_1;\xi_2}$.

- $\xi = \xi_1^*$: Let Z be the least set containing X and closed under the condition: for all w, if $\widehat{w} \wedge \langle \xi_1 \rangle^{\forall} \widetilde{Z}$ is consistent, then $w \in Z$. By definition of Z and induction hypothesis, we get for all $w \in Z$, there exists $X_w \subseteq X$ such that $(w, X_w) \in R_{\xi_1^*}$.

  Claim : $\vdash \langle \xi_1 \rangle^{\forall} \widetilde{Z} \supset \widetilde{Z}$.

  To see the claim, suppose it is not true. Then $\langle \xi_1 \rangle^{\forall} \widetilde{Z} \wedge \neg \widehat{Z}$ is consistent. Let $Z' = AT(\alpha_0) \setminus Z$. We have $\langle \xi_1 \rangle^{\forall} \widetilde{Z} \wedge \widetilde{Z'}$ is consistent. Therefore there exists $w' \in Z'$ such that $\langle \xi_1 \rangle^{\forall} \widetilde{Z} \wedge \widehat{w'}$ is consistent. But then w' would have been added into the set Z during construction.

  End of claim

Applying the induction rule (IND), we have $\vdash \langle \xi_1^* \rangle^\forall \widetilde{Z} \supset \widetilde{Z}$. By assumption, $\widehat{u} \wedge \langle \xi_1^* \rangle^\forall \widetilde{X}$ is consistent. So $\widehat{u} \wedge \langle \xi_1^* \rangle^\forall \widetilde{Z}$ is consistent. Hence $\widehat{u} \wedge \widetilde{Z}$ is consistent and therefore $u \in Z$. Thus we have $(u, X') \in R_{\xi_1^*}$ for some $X' \subseteq X$.

□

Lemma 4.2.15 *For all $\langle \xi \rangle^\forall \alpha \in CL(\alpha_0)$, for all $u \in W$, $\widehat{u} \wedge \langle \xi \rangle^\forall \alpha$ is consistent iff there exists $(u, X) \in R_\xi$ such that $\forall w \in X$, $\alpha \in w$.*

Proof: ($\Rightarrow$) Suppose $\widehat{u} \wedge \langle \xi \rangle^\forall \alpha$ is consistent. Let $X_\alpha = \{w \in W \mid \alpha \in w\}$. It is easy to see that $\vdash \alpha \supset \widetilde{X}_\alpha$. Therefore we have $\widehat{u} \wedge \langle \xi \rangle^\forall \widetilde{X}_\alpha$ is consistent (by definition $\forall w \in \widetilde{X}_\alpha$, $\alpha \in w$). By Lemma 4.2.14, there exists $X' \subseteq X_\alpha$ such that $(u, X') \in R_\xi$. Since $X' \subseteq X_\alpha$, we have $\alpha \in w$ for all $w \in X'$.
($\Leftarrow$) Suppose $\exists (u, X) \in R_\xi$ such that $\forall w \in X$, $\alpha \in w$. We need to show that $\widehat{u} \wedge \langle \xi \rangle^\forall \alpha$ is consistent, this is done by induction on the structure of ξ.

- The case when $\xi = (g, \sigma)$ follows easily from Lemma 4.2.13 and $\xi = \xi_1 \cup \xi_2$ follows from the induction hypothesis and axiom (A3a).

- $\xi = \xi_1; \xi_2$: Since $(u, X) \in R_{\xi_1;\xi_2}$, there exist $Y = \{v_1, \ldots, v_k\}$ and sets $X_1, \ldots, X_k \subseteq X$ such that $\bigcup_{j=1,\ldots,k} X_j = X$, for all $j : 1 \leq j \leq k$, $(v_j, X_j) \in R_{\xi_2}$ and $(u, Y) \in R_{\xi_1}$. By induction hypothesis, for all j, $\widehat{v_j} \wedge \langle \xi_2 \rangle^\forall \alpha$ is consistent. Since v_j is an atom and $\langle \xi_2 \rangle^\forall \alpha \in CL(\alpha_0)$, we get $\langle \xi_2 \rangle^\forall \alpha \in v_j$. Again by induction hypothesis we have $\widehat{u} \wedge \langle \xi_1 \rangle^\forall \langle \xi_2 \rangle^\forall \alpha$ is consistent. Hence from (A3b) we have $\widehat{u} \wedge \langle \xi_1; \xi_2 \rangle^\forall \alpha$ is consistent.

- $\xi = \xi_1^*$: If $u \in X$ then $\vdash \widehat{u} \supset \widetilde{X}$. We have $\vdash \widetilde{X} \supset \alpha$ and hence we get $\widehat{u} \wedge \alpha$ is consistent. From axiom (A3c) we have $\widehat{u} \wedge \langle \xi_1^* \rangle^\forall \alpha$ is consistent.

  Else we have $(u, X) \in R_{\xi_1;\xi_1^*}$. Let $Z_0 = X$ and $Z_{n+1} = Z_n \cup \{w \mid (w, Z') \in R_{\xi_1}, Z' \subseteq Z_n\}$. Take the least m such that $u \in Z_m$. We have for all $w \in Z_{m-1}$, $\vdash \widehat{w} \supset \langle \xi_1^* \rangle^\forall \widetilde{X'}$ for some $X' \subseteq X$. We also have $(u, Z'_m) \in R_{\xi_1}$ for some $Z'_m = \{v_1, \ldots, v_k\} \subseteq Z_m$. Let $X_1, \ldots, X_k \subseteq X$ such that $\forall j : 1 \leq j \leq k$, we have $(v_j, X_j) \in R_{\xi_1^*}$ and $X' = \bigcup_{j=1,\ldots,k} X_j$. By an argument similar to the previous case we can show that $\widehat{u} \wedge \langle \xi_1 \rangle^\forall \langle \xi_1^* \rangle^\forall \widetilde{X'}$ is consistent. Hence we get $\widehat{u} \wedge \langle \xi_1; \xi_1^* \rangle^\forall \alpha$ is consistent. Therefore from axiom (A3c) we have $\widehat{u} \wedge \langle \xi_1^* \rangle^\forall \alpha$ is consistent.

□

We thus get the following theorem which implies the completeness of the axiom system.

Theorem 4.2.16 *For all $\beta \in CL(\alpha_0)$, for all $u \in W$, $M, u \models \beta$ iff $\beta \in u$.*

Proof: Follows from Lemma 4.2.15 by a routine inductive argument. □

Decidability: Since the size of the action set $|\Sigma|$ is constant, the size of $CL(\alpha_0)$ is linear in $|\alpha_0|$. Atoms are maximal consistent subsets of $CL(\alpha_0)$, hence $|AT(\alpha_0)|$ is exponential in the size of α_0. From the completeness theorem we get that for a formula α_0, if α_0 is satisfiable then it has a model of exponential size, i.e. $|W| = \mathcal{O}(2^{|\alpha_0|})$. For all game strategy pairs ξ occurring in α_0, the relation R_ξ can be computed in time exponential in the size of the model. Therefore it follows that the logic is decidable in nondeterministic double exponential time.

4.2.6 Extensions

Concurrency as introduced in game logic [BGL07] can be represented in our framework with the addition of the operator $\xi_1 \times \xi_2$ in the syntax of game strategy pairs. For instance, $(g_1, \sigma_1) \times (g_2, \sigma_2)$ would mean that the game g_1 is played with a strategy conforming to σ_1 and concurrently, the game g_2 is played with a strategy conforming to σ_2. The semantics can be defined in the usual manner:

- $R_{\xi_1 \times \xi_2} = \{(u, X) \mid X = X_1 \cup X_2$ such that $(u, X_1) \in R_{\xi_1}$ and $(u, X_2) \in R_{\xi_2}\}$.

It is easy to see that the completeness theorem also follows with the addition of the following axiom.

- $\langle \xi_1 \times \xi_2 \rangle \alpha \equiv \langle \xi_1 \rangle \alpha \wedge \langle \xi_2 \rangle \alpha$.

Test operator

The test operator as in dynamic logic can also be added into the syntax of game strategy pairs. For $\beta \in \Phi$, the interpretation of $\beta? \in \Gamma$ would be to test whether β holds at the particular state and if yes, continue else fail. The semantics can be given as:

- $R_{\beta?} = \{(u, \{u\}) \mid M, u \models \beta\}$.

The test operator gives the ability of checking for certain conditions and then deciding which game to proceed with. This construct is particularly interesting in our framework, since unlike programs we have players in the game. For instance, let π denote the strategy specification of player 2 and σ the specification of player 1. The formula $(g_1, \pi); win_2?; (g_2, \sigma)$ says that in g_1 if player 2 by employing a strategy conforming to π can ensure win_2 then proceed with the game g_2 where player 1 plays σ. Note that if the test fails then g_2 is not played. This is in contrast to the tests performed in a strategy specification. In a specification if the test fails then the player is free to choose any action.

With the addition of the following axiom, the completeness theorem goes through.

- $\langle \beta? \rangle \alpha \equiv \beta \wedge \alpha$

4.3 Normal form games

As opposed to extensive form games where the game structure is explicit, normal form games are specified by the set of abstract strategies and outcomes. Logical analysis in the case of a single normal form game is thus outcome based. However when we consider games built in a compositional manner, the notion of strategic response of a player to other players' moves become relevant, pretty much in the same way as it is used in extensive form games.

In this section we look at how the dynamic logic framework developed to reason about extensive form games can be adapted to deal with normal form games as well. We consider composition of game play pairs in normal form games, corresponding to the fact that the reasoning performed in single stage is mostly outcome based. If we restrict the reasoning to bounded repetition of games or to multistage games where the number of stages are bounded, then we do not need to look at composition of game play pairs. In the presence of unbounded iteration of games, we need to introduce a dynamic structure on game play pairs.

4.3.1 Syntax for normal form games

Normal form games were introduced in Section 2.2 and we consider the tree representation for normal form games. Recall that for each $i \in N$, Σ^i denotes the actions

of player i and $\widehat{\Sigma} = \Sigma^1 \times \Sigma^2$. Let *Nodes* be a countable set, the normal form game tree is specified using the syntax:

$$\mathbb{G}(Nodes) := \Sigma_{a_m \in J}(x, a_m, y_m).$$

where $x, y_m \in Nodes$, $J \subseteq \widehat{\Sigma}$.

Given $g \in \mathbb{G}(Nodes)$ we define the tree T_g generated by g inductively as follows. Let $g = (x, a_1, y_1) + \ldots + (x, a_k, y_k)$, $T_g = (S_g, \Rightarrow_g, \widehat{\lambda}_g, s_x)$ where

- $S_g = \{s_x, s_{y_1}, \ldots, s_{y_k}\}$.
- For $1 \leq j \leq k$ we have $s_x \overset{a_j}{\Rightarrow}_g s_{y_j}$.

For $g \in \mathbb{G}(Nodes)$, we also use $\widehat{\Sigma}^g$ to denote the set of all strategy profiles in g.

4.3.2 Dynamic logic on normal form games

For the sake of clarity, in this section we concentrate on the structure of the game g with respect to the moves of the players and disregard the utilities associated in the game structure. As shown in Section 2.4 utilities can be coded as propositions in the logic and thus outcome based reasoning can be done.

Syntax: The syntax of the logic is given by,

$$\Phi := p \in P \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle\xi\rangle^{\forall}\alpha$$

where $\xi \in \Gamma$. The syntax of game play pairs is given as

$$\Gamma := (g, \eta) \mid \xi_1; \xi_2 \mid \xi_1 \cup \xi_2 \mid \xi^* \mid \alpha?$$

where $g \in \mathbb{G}(Nodes)$, $\eta \subseteq \Sigma^g$ and $\alpha \in \Phi$.

Semantics: Models for the logic are Kripke structures $M = (W, \rightarrow, V)$. Note that unlike in the case of extensive form games, the turn function is not required.

The truth of a formula $\alpha \in \Phi$ in the model M at a position w (denoted $M, w \models \alpha$) is defined as follows:

- $M, w \models p$ iff $p \in V(w)$.
- $M, w \models \neg\alpha$ iff $M, w \not\models \alpha$.

- $M, w \models \alpha_1 \vee \alpha_2$ iff $M, w \models \alpha_1$ or $M, w \models \alpha_2$.
- $M, u \models \langle \xi \rangle^{\forall} \alpha$ iff $\exists (u, X) \in R_\xi$ such that $\forall w \in X$ we have $M, w \models \alpha$.

For $\xi \in \Gamma$ the definition of relation $R_\xi \subseteq W \times 2^W$ is similar to what we saw in the case of extensive form games. For the atomic case we have,

- $R_{(g,\sigma)} = \{(u, X) \mid \mathit{enabled}(g, u) \text{ and } X = \mathit{tail}(T_u \restriction g, \eta)\}$

where $\mathit{enabled}(g, w)$ denotes that the structure g can be embedded at state w of the model, with respect to compatibility with the action labels. $\mathit{tail}(T_u \restriction g, \eta)$ is the set of nodes of the resulting embedded tree when restricted to plays in η.

Thus $M, w \models \langle g, \eta \rangle^{\forall} \alpha$ says that firstly g can be embedded at u and if X is the set of all states resulting from the plays specified in η, then the formula α holds in all $w \in X$. The dual $[g, \eta]^{\exists} \alpha$ says: if g can be embedded at the state u then there exists a state w resulting from the plays specified in η such that α holds at w.

The semantics for composite game play pairs is given as follows:

- $R_{\xi_1;\xi_2} = \{(u, X) \mid \exists Y \subseteq W$ such that $(u, Y) \in R_{\xi_1}$ and $\forall v \in Y$ there exists $X_v \subseteq X$ such that $(v, X_v) \in R_{\xi_2}$ and $\bigcup_{v \in Y} X_v = X\}$.
- $R_{\xi_1 \cup \xi_2} = R_{\xi_1} \cup R_{\xi_2}$.
- $R_{\xi^*} = \bigcup_{n \geq 0} (R_\xi)^n$.
- $R_{\beta?} = \{(u, \{u\}) \mid M, u \models \beta\}$.

Example 4.3.1 The formulas of the logic can not only make assertions about strategies of players but also about the game structure itself. Thus states of the Kripke structure can be viewed as being associated with a set of atomic normal form games. The restriction operation identifies the specific game under consideration, which in turn is determined by the assertions made by formulas of the logic. Consider the following formula:

- $\langle (g, \eta_2); (g', \eta_1) \rangle^{\forall} \mathit{win}_1$ where η_2 is a strategy for player 2 in game g and η_1 a strategy of player 1 in g'.

This says that assuming in game g, player 2 plays according to strategy η_2 then in g', player 1 can follow η_1 and ensure win_1. Note that this is not same as saying player 1 can ensure win_1 in the composed game $g = g; g'$. The fact that player 2 employed strategy η_2 in game g is used in strategizing by player 1. □

4.3.3 Logical reasoning in normal form games

Models of alternating temporal logic can also be thought of as games on graphs where each node is associated with a normal form game. Thus specifications which involve only bounded levels of strategic response (as shown in the previous example), can be expressed in a temporal logic framework. Consider the following assertion.

- $((g_1, \eta_1); ((g_2, \eta_2) \cup (g_3, \eta_3)))^*; win_2?; (g, \eta)$
  where η_1, η_2 and η_3 are player 2 strategies in games g_1, g_2 and g_3 respectively and η is a player 1 strategy in game g.

This says that if player 2 can ensure win_2 by iterating the structure g_1 followed by g_2 or g_3 and employing strategies η_1 followed by η_2 or η_3 then player 1 plays according to η in game g. Here not only does player 1 assert that player 2 can ensure win_2 but also makes assertions about the specific game structure that is enabled and the atomic strategies that player 2 employs. Iteration performed here does not correspond to the assertion that a property holds through out the history. This also motivates the need to shift from a temporal logic framework to a dynamic logic framework.

Strategy comparison: We now show that the logic is powerful enough to express the various strategizing notions including strategy comparison for reasoning about a single normal form game. For the game g, let $\widehat{\Sigma}^g = \{a_1, \ldots, a_k\}$ be the strategy profiles occurring in g. For $i \in N$, let $\Sigma_i^g = \{a_1[i], \ldots, a_k[i]\}$ and for $b \in \Sigma_i^g$, let $\widehat{\Sigma}_g(b) = \{a \in \widehat{\Sigma}^g \mid a[i] = b \text{ and } a[\overline{i}] \in \Sigma_{\overline{i}}^g\}$. $\widehat{\Sigma}_g(b)$ thus consists of all the strategy profiles where player i's strategy is fixed to b. Consider the formula:

$$ensures^i(g, \gamma) = \bigvee_{b \in \widehat{\Sigma}_i^g} \langle g, \widehat{\Sigma}_g(b) \rangle^{\forall} \gamma.$$

$ensures^i(g, \gamma)$ says that given that the opponent chooses an action from the set $\Sigma_{\overline{i}}^g$, there is a strategy for player i to achieve γ no matter what choice player $\overline{i}$ makes. In the case of $\gamma \in \Theta_i$ (where Θ_i denote the set of special propositions coding the utilities of players), this corresponds to the utility that player i can ensure. If player i expects that $\overline{i}$ will choose only actions from the set $\Sigma' \subseteq \Sigma_{\overline{i}}^g$, then the restriction of $ensures^i(g, \gamma)$ to Σ' specifies what player i can ensure in terms of his expectation. A player during the phase of strategizing might take into consideration what he

can ensure given his expectation about the strategies of the opponent. The related concept of weakly dominating strategies can be defined as follows:

$$DOM^i(b,b') = \bigwedge_{x \in \Sigma^g_{\overline{i}}} \bigwedge_{\theta_i \in \Theta_i} \left(\langle g,(b',x)\rangle^{\forall}\theta_i \supset \langle g,(b,x)\rangle^{\forall}\theta_i \right).$$

This says that whatever reward that can be ensured using the strategy b' can also be ensured with the strategy b. In other words, this says that for player i, the strategy b weakly dominates b'.

Given a strategy x of player $\overline{i}$ we can express the fact that the strategy b is better than b' for player i as response to x using the formula:

$$Better^i_x(b,b') = \bigwedge_{\theta_i \in \Theta_i} (\langle g,(b',x)\rangle^{\forall}\theta_i \supset \langle g,(b,x)\rangle^{\forall}\theta_i)$$

We can then express the fact that b is the best response of player i for x as $BR^i_x(b) = \bigwedge_{b' \in \Sigma^g_i} Better^i_x(b,b')$. Having defined best response, the assertion that a strategy profile (b,x) constitutes an equilibrium can be expressed as: $EQ(b,x) = BR^i_x(b) \wedge BR^{\overline{i}}_b(x)$.

Capturing complete strategies: Typically temporal logics which allow strategies to be named and referred to in the logic restrict attention to memoryless strategies and consider them to be atomic and unstructured. The reasoning performed is in terms of outcome based analysis. If we restrict our attention to memoryless strategies, then complete strategies of players can be coded up in terms of strategy specifications. We need to only use special propositions to distinguish each state of the arena. Thus the core reasoning done in such temporal logics is subsumed by the dynamic logic on normal form games. In addition to outcome based analysis, the logic proposed here explicates strategic response of players in terms of the mechanisms which need to be employed.

4.3.4 Axiom system and completeness

In this section we show that the axiom system presented in Section 4.2.4 for extensive form games can be easily adapted for normal form games. The axioms which need to be modified are the ones dealing with the atomic case which reflect the change in the underlying game representation. At the compositional level, the axioms remain the same.

As in the case of extensive form games, the first step is to show that for $a \in \widehat{\Sigma}$, the standard modal logic formula $\langle a \rangle \alpha$ can be encoded in the logic. For $a \in \widehat{\Sigma}$, let g_a denote the normal form game with the unique strategy profile a, we define $\langle a \rangle \alpha$ as:

- $\langle a \rangle \alpha = \langle g_a, \{a\} \rangle^{\forall} True \wedge [g_a, \{a\}]^{\exists} \alpha$.

The following proposition can be easily verified.

Proposition 4.3.2 *For all models M, state w and formula α, $M, w \models \langle a \rangle \alpha$ iff there is a state u such that $w \xrightarrow{a} u$ and $M, u \models \alpha$.*

For a game $g = (x, a_1, y_1) + \ldots + (x, a_k, y_k)$, the formula $g^{\checkmark}$ denotes that the game structure g is enabled. This is defined as:

- $g^{\checkmark} = \bigwedge_{j=1,\ldots,k} \langle a_j \rangle True$.

The axiom schemes

(A1) Propositional axioms:

(a) All the substitutional instances of tautologies of PC.

(A2) Axiom for single edge games:

(a) $\langle a \rangle (\alpha_1 \vee \alpha_2) \equiv \langle a \rangle \alpha_1 \vee \langle a \rangle \alpha_2$.

(A3) Dynamic logic axioms:

(a) $\langle \xi_1 \cup \xi_2 \rangle^{\forall} \alpha \equiv \langle \xi_1 \rangle^{\forall} \alpha \vee \langle \xi_2 \rangle^{\forall} \alpha$.

(b) $\langle \xi_1 ; \xi_2 \rangle^{\forall} \alpha \equiv \langle \xi_1 \rangle^{\forall} \langle \xi_2 \rangle^{\forall} \alpha$.

(c) $\langle \xi^* \rangle^{\forall} \alpha \equiv \alpha \vee \langle \xi \rangle^{\forall} \langle \xi^* \rangle^{\forall} \alpha$.

(d) $\langle \beta ? \rangle^{\forall} \alpha \equiv \beta \wedge \alpha$.

For $g = (x, a_1, y_1) + \ldots + (x, a_n, y_n)$ and $\eta \subseteq \widehat{\Sigma}^g$,

(A4) $\langle g, \eta \rangle^{\forall} \alpha \equiv g^{\checkmark} \wedge (\bigwedge_{a \in \eta} [a] \alpha)$.

Inference rules

$$(MP)\ \frac{\alpha, \quad \alpha \supset \beta}{\beta} \qquad (NG)\ \frac{\alpha}{[a]\alpha}$$

$$(IND)\ \frac{\langle \xi \rangle^{\forall} \alpha \supset \alpha}{\langle \xi^* \rangle^{\forall} \alpha \supset \alpha}$$

The soundness of the axiom system and inference rules can be verified quite easily. Using arguments very similar to those presented in Section 4.2.5 it can be shown that given a consistent formula α_0 we can construct a model $M = (W, \rightarrow, V)$ where W consists of atoms of α_0 such that the following holds:

Theorem 4.3.3 *For all $\beta \in CL(\alpha_0)$, for all $u \in W$, $M, u \models \beta$ iff $\beta \in u$.*

Completeness of the axiom system can in turn be derived from Theorem 4.3.3.

4.4 Discussion

By considering game play pairs in the case of structured normal form games, we are able to reason about restrictions of the game tree and thereby express game theoretic notions like a player's best response for an opponent's strategy and equilibrium. In contrast, the approach taken in the case of extensive form games is closer to the style of game logics: the reasoning is about what a player can ensure by following a certain strategy specification where all possible strategies of the opponent is taken into account. However, at the compositional level, the axiom system remains the same. This shows that the framework being considered is quite general, and is not dependent on the exact game representation. For a specific game representation, only the axioms specifying the structure of the representation need be changed.

Part II

Algorithmic analysis

Chapter 5

Infinite games

Consider the game played between two players, Player 1 and Player 2, who take turns to choose binary digits. Player 1 makes the first choice and then their turns strictly alternate. Let X be a subset of the real open interval $(0, 1)$. A play in the game is a sequence $x_1, y_1, x_2, y_2, \ldots$ where for all $j \geq 0$, $x_j, y_j \in \{0, 1\}$ and let $\mathfrak{n} = \frac{x_1}{2} + \frac{y_1}{2^2} + \frac{x_2}{2^3} + \cdots$. We say player 1 wins if $\mathfrak{n} \in X$ and player 2 wins if $\mathfrak{n} \notin X$. It is easy to see that this defines an infinite two player turn based game of perfect information. This game was first defined in *The Scottish Book* [Mau81] by Ulam. A closely related variant of this game is called the Banach-Mazur game (see [Mau81] for the original variant) where players are allowed to choose intervals on the real line. These games have been extensively studied in descriptive set theory [Kec95] and topology [Tel87]. In general, given an alphabet set A and a set $X \subseteq A^\omega$ of infinite words on A we can define a two player zero sum game $G(X)$ where players alternate in choosing elements from A constructing an infinite sequence $\mathfrak{n}$. Player 1 wins if the resulting sequence $\mathfrak{n} \in X$ and player 2 wins if $\mathfrak{n} \notin X$. The obvious question of interest then is to ask whether such infinite games are determined. It turns out that determinacy depends crucially on the topological properties of the winning set X. Under topological classifications, one of the simplest games to consider would be open games (a game is said to be open if the winning set has the form $X = UA^\omega$). In 1953, Gale and Stewart [GS53] showed that any open game is determined. Various studies extended this result to larger classes of infinite games. However, it was not until 1975 that determinacy was shown for a very general class of games; the celebrated Martin's theorem [Mar75] showed that all Borel games are determined.

Infinite games have been long used in various aspects of computer science (see [GTW02, Grä08] for an overview). It has been widely used in automata theory

to show the closure under complementation for various classes of automata as well as in deciding the emptiness problem. For instance, for a nondeterministic tree automaton working on infinite trees, there is a natural two player zero sum game of perfect information associated with it. In the game, the automaton picks an enabled transition, and the opponent chooses a branch to pursue on the input tree. Then the complementation problem for this class of automata is solved by using determinacy of the associated games. Infinite games are often used in the verification and synthesis of open systems. For instance, Church's problem [Chu63] which asks whether it is possible to synthesize circuits against specifications stated in restricted second-order arithmetic can be easily translated into the determinacy question for an infinite game.

Typically, infinite games which arise in computer science are games played on finite graphs with regular objectives. These objectives fall in the second level of the Borel hierarchy [PP04] and thus determinacy for such games follows from Martin's theorem. However, Martin's result does not make any assertion on whether it is possible to determine who the winner is or how "complex" the winning strategy is. This turns out to be the core question in solving verification and synthesis questions as well.

Apart from the logical analysis, another branch of game theory which is of particular interest in the context of computer science is the algorithmic analysis of games and strategies. This encompasses issues mentioned above including,

- being able to determine the winner and synthesizing the winning strategy in the case of two player zero sum games, and
- synthesis of equilibrium strategy profile in non-zero sum games.

Algorithmic analysis for finite extensive form games, was briefly looked at in Chapter 2 where we presented the backward induction algorithm. The algorithm showed that winning strategy synthesis and equilibrium strategy profile synthesis can be achieved in time linear in the size of the game tree. In the case of two player infinite games with regular objectives, being able to determine the winner as well as computing the winning strategy also turn out to be the core questions in solving the verification and synthesis problems. However, since the backward induction algorithm is designed to work on finite extensive form game trees, this procedure cannot be directly applied in the analysis of infinite duration games. A seminal result due to Büchi and Landweber [BL69] says that for two player zero

sum games played on finite graphs where players' objectives are presented as Muller conditions, the winner can be determined and that the winning strategy can be effectively synthesised in finite memory strategies.

In the context of non-zero sum infinite duration games, it is natural to ask if equilibrium strategies exist and whether it is possible to synthesize an equilibrium profile if it exists. We show that in the case where preference orderings of players are over regular sets of plays, the backward induction procedure can still be employed to show the existence of equilibrium and to synthesize an equilibrium profile. We also look at how strategy specifications help in the algorithmic analysis of non-zero sum infinite duration games.

5.1 Game model

In this chapter we consider infinite turn based games with perfect information played on finite graphs. We use game arenas (introduced in section 2.3.1) to represent such games. For technical convenience we also assume that for a game arena $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ for all game positions $w \in W$, we have $\vec{w} \neq \emptyset$. Strategies of players can be defined as given in section 2.3.2.

5.1.1 Objectives of players

Two player zero sum games

For two player zero sum games, the objectives of players are strictly complementary. Thus the set of plays need to be partitioned into sets Φ_1 and Φ_2 with the interpretation that a play ρ is winning for player i iff $\rho \in \Phi_i$. Since the game is strictly complementary, this also means that ρ is losing for player $\overline{i}$. A strategy μ is winning for player i if for all paths $\rho \in T_\mu$, $\rho \in \Phi_i$. A game G is then specified as a tuple $(\mathcal{G}, \Phi_1)$ where $\mathcal{G}$ is an arena and Φ_1 denotes the winning condition of one of the players (say player 1). For $i \in \{1, 2\}$ we say player i wins the game G if i has a winning strategy in G. The game G is said to be determined if there exists $i \in \{1, 2\}$ such that i wins G.

For two player zero sum infinite games played on finite graphs, the algorithmic questions of interest include given a game $G = (\mathcal{G}, \Phi_1)$

1. is G determined?

2. given i, determine if player i wins G and if so compute the winning strategy.

However, for algorithmic analysis to be possible, we need to present the winning conditions of players in a finite fashion. Since the objectives of players is an infinite set it is not clear how this can be done in general. Most often infinite games which arise in computer science turn out to be two player zero sum games with regular objectives. In such games, the winning conditions of players can be presented in a finite manner in terms of omega automata. Below we illustrate how this can be achieved.

Definition 5.1.1 *A finite deterministic omega automaton over the input alphabet $W \times \Sigma$ is a tuple $\mathbb{A} = (R, \Delta, r_0, Acc)$ where*

- *R is the set of states.*
- *$\Delta : R \times W \times \Sigma \to R$ is the transition function.*
- *$r_0 \in R$ is the initial state.*
- *Acc specifies the acceptance condition.*

The run of $\mathbb{A}$ on an infinite sequence $\rho : w_0 a_0 w_1 \ldots$ is a sequence of states $\varphi_\rho : r_0 r_1 \ldots$ such that for all $j \geq 0$, $r_{j+1} = \Delta(r_j, w_j, a_j)$. Let $Inf(\varphi_\rho)$ denote the set of states occurring infinitely often in φ. The most commonly used acceptance conditions are the following requirements on $Inf(\varphi)$:

- Büchi condition [Büc62]: for a set of "good states" $\mathcal{B} \subseteq R$, $Inf(\varphi_\rho) \cap \mathcal{B} \neq \emptyset$. In other words, some final state occurs infinitely often in the run φ_ρ.
- Muller condition [Mul63]: for a family $\mathcal{F} \subseteq 2^R$, $\bigvee_{F \in \mathcal{F}} Inf(\varphi_\rho) = F$. This requires that the set of states occurring infinitely often in the run φ_ρ forms a set in $\mathcal{F}$.
- Rabin condition [Rab69]: For a set of pairs $\{(E_j, F_j)\}_{j=1,\ldots,m}$ where $E_j, F_j \subseteq R$, we have $\bigvee_{j=1}^{m} (Inf(\varphi_\rho)) \cap E_j = \emptyset \wedge Inf(\varphi_\rho)) \cap F_j \neq \emptyset$. It requires that for some j, all states of E_j are visited only finitely often in φ_ρ but some state of F_j is visited infinitely often.

Deterministic Muller automata are known to be complete for omega regular conditions and therefore we assume that winning conditions of players are presented in

this manner. We use the notation $\mathcal{M} = (R, \Delta, r_0, \mathcal{F})$ to denote a Muller automaton and always work with deterministic automata unless otherwise mentioned.

A two player zero sum game with omega regular objectives can thus be presented as a pair $G = (\mathcal{G}, \mathcal{M}_1)$ where $\mathcal{M}_1$ specifies the winning condition for player 1.

Overlapping objectives

For non-zero sum games, each player has a preference ordering $\preceq^i \subseteq (\mathit{Plays}(\mathcal{G}) \times \mathit{Plays}(\mathcal{G}))$ over plays in the arena. The most natural way of specifying the preference ordering is in terms of utilities as we did in the case of finite extensive form games. However, since plays in the arena are infinite objects, the utility function needs to map infinite plays to payoffs. If we restrict our attention to classifying regular plays and in cases where the utilities arise out of a finite set, players' objectives can be presented in terms of a "generalised Muller automaton" which we term as evaluation automata.

Evaluation automata: These are basically Muller automata where instead of interpreting the Muller table as defining accepting runs we incorporate preference orderings over the sets in the Muller table. The ordering on plays induced by the utility function can be directly captured in this manner. The formal definition is given below.

Definition 5.1.2 *An evaluation automaton $\mathcal{E} = (\mathcal{M}, \{\lhd^i\}_{i \in N})$ where $\mathcal{M}$ is a Muller automaton given by $\mathcal{M} = (R, \Delta, r_0, \mathcal{F})$ and for each player $i \in N$, $\lhd^i \subseteq (\mathcal{F} \times \mathcal{F})$ is a reflexive, transitive and complete relation over $\mathcal{F}$ denoting the preference ordering.*

Since we want the evaluation automaton to induce a preference ordering on all plays, it is convenient to use a "complete" automaton. Every evaluation automaton can be converted into a complete automaton by the following transformation. For the Muller automaton $\mathcal{M} = (R, \Delta, r_0, \mathcal{F})$ we construct the automaton $\mathcal{M}' = (R, \Delta, r_0, \mathcal{F}')$ where $\mathcal{F}' = 2^R \setminus \emptyset$. The newly added final states are set to be the least preferred by each player i in the preference ordering $\lhd^i$. Therefore without loss of generality we assume that all evaluation automata are complete.

The run of $\mathcal{E}$ on a play ρ (denoted φ_ρ) is defined as in the case of a Muller automaton. The evaluation automaton $\mathcal{E}$ induces a preference ordering on $\mathit{Plays}(\mathcal{G})$ in the following manner. Let $\rho = w_0 a_0 w_1 a_1 \ldots$ and $\rho' : w_0 a'_0 w'_1 a'_1 \ldots$ be two plays. For player $i \in N$, we have $\rho \preceq^i \rho'$ iff $\mathit{Inf}(\varphi_\rho) \lhd^i \mathit{Inf}(\varphi_{\rho'})$.

We shall also be interested in the special case of binary evaluation automata which specify least outcomes for player i. Such an automaton is given by $\mathcal{E}^i_F$, where $F \in \mathcal{F}$: for every $F' \in \mathcal{F}$, if $F \lhd^i F'$, it is taken to be "winning" for player i, and every $F'' \neq F$ such that $F'' \lhd^i F$ is taken to be "losing". Such an automaton checks if i can ensure an outcome which is at least as preferred as F. Note that the terminology of win-loss is only to indicate a binary preference for player i, and applies even in the context of non-zero sum games.

Nash equilibrium: The definitions of best response strategy and equilibrium profile can be appropriately modified to deal with infinite duration games on graphs in the following manner.

- The strategy μ of player i is a best response for strategy τ (of $\overline{i}$) if $\forall \mu' \in \Omega^i$, $\rho_{(\mu',\tau)} \preceq^i \rho_{(\mu,\tau)}$.
- A strategy profile (μ, τ) is said to be in equilibrium if μ is the best response for τ and τ is the best response for μ.

Equilibrium in win-loss objectives

In the context of equilibrium computation for infinite games, instead of looking at general preference orderings over outcomes, one could look at the situation where the objective of each player is specified as an omega regular win-loss objective. These sets may overlap and hence the players need not be antagonistic. In other words, these are non-zero sum games where each player has a binary objective. The existence of Nash equilibrium for such games follows from the result of [CJM04]. The main idea here is the effective use of threat strategies whereby a player deviating from the equilibrium profile is punished by others to receive the outcome which she can guarantee on her own. The existence of sub-game perfect equilibrium [Sel65] for games with binary objectives was shown in [Umm05]. Threat strategies arise naturally in the case of games with win-loss objectives. However, in the case of infinite games where players have non-zero sum objectives, it is no longer clear what are the rationality assumptions which justify equilibrium profiles that may involve empty threats. In this context, even coming up with rationality assumptions which generalise well-known solution concepts in games of finite duration to that of non-zero sum infinite duration games is a challenging task. [Ber07] takes up the task of looking at the rationality assumptions involved in generalising the notion of iterated

admissibility [LR57] which is well studied in the theory of finite games to infinite games.

However, the equilibrium notions are mathematically well defined and deserves attention in their own right. What we do here is rather than delve into issues concerning rationality, attempted to investigate equilibrium notions in the context of infinite games. In the next section we show that the standard technique of backward induction can be appropriately modified to compute equilibrium profile in generalised Muller games.

5.2 Equilibrium computation

Given a game arena and an evaluation automaton, their product gives rise to a Muller game where each player has a preference ordering over the connected components over the product structure. We call these generalised Muller games (the formal definition is presented shortly). A natural question then, would be to ask whether an equilibrium profile always exists for this class of games. In this section we show the following results:

- Nash equilibrium always exists in generalised Muller games.
- An equilibrium profile can be effectively synthesized.

A generalised Muller game is a tuple $G = (\mathcal{G}, \{\sqsubseteq^i\}_{i \in N})$ where $\mathcal{G}$ is a game arena with the set of game positions W and for each player $i \in N$, $\sqsubseteq^i \subseteq (2^W \times 2^W)$ is a preference ordering over subsets of game positions for each player.

To simplify notation, we disregard the action labels on edges of the arena. Thus players' strategies choose game positions instead of actions. We further assume that the turn function is implicitly presented by a partition of the game positions. Thus an arena is simply a graph $\mathcal{G} = (W, E, w_0)$ where $W = \bigcup_{i \in N} W^i$ is the set of game positions partitioned into $|N|$ sets. The move relation $E \subseteq W \times W$. Let $wE = \{w' \mid (w, w') \in E\}$.

5.2.1 Nash equilibrium in generalised Muller games

For finite extensive form games, our main tool for algorithmic analysis was the backward induction procedure. Here we show that the backward induction procedure can

be effectively used to show existence of equilibrium and to synthesize an equilibrium strategy profile for generalised Muller games as well.

The key idea of the construction is as follows. The backward induction algorithm is designed to work on finite extensive form game trees. However, the tree unfolding of G which is the game under consideration is an infinite structure. The objective is therefore to construct a finite tree structure which preserves the equilibrium behaviour of players in G. The core problem in constructing such a structure is to identify the Muller set that each play settles down to without actually performing the infinite tree unfolding. In the context of omega automata, the data structure which combines the latest appearance record [GH82] along with a hit position was proposed by Büchi [Büc83] with exactly this purpose in mind. We show that by performing a careful unfolding of G while keeping track of the permutation of states in terms of the latest appearance record, one can construct a finite tree structure $T(G)$ which captures the equilibrium behaviour of players in the original game G. The backward induction procedure synthesizes a memoryless equilibrium profile on the tree structure $T(G)$ which is then translated into a finite memory equilibrium profile in G.

Latest appearance record

The idea is to keep a record of states in the order of their "last visit" along with a hit position (denoted by the symbol $\sharp$) which records the position of the last change. We introduce the data structure by an example, the formal definition is presented subsequently.

Example 5.2.1 Suppose $W = \{1, 2, 3, 4\}$, consider the infinite sequence 1 4 2 3 1 2 1 2 2 1 ... over W which finally loops in the set $\{1, 2\}$. We start with a vector whose last state is 1 say 234$\sharp$1 indicating that the sequence begins with 1. The next vector is obtained by shifting the new state of W to the right and setting the hit to the position from where the previous vector this state was taken. Thus we obtain, starting from (234$\sharp$1) the vectors (23$\sharp$14), ($\sharp$3142), ($\sharp$1423), ($\sharp$4231), (4$\sharp$312), (43$\sharp$21) and so on. It is easy to see that in this example, where from some point onwards only states 1,2 are visited, these states remain at the positions of the vector after the $\sharp$ symbol. □

Formally, given a finite set W which is well ordered, we define $\mathbb{LAR}(W)$ as follows:

$$\mathbb{LAR}(W) = \{x \in (W \cup \{\sharp\})^* \mid \forall v \in W \cup \{\sharp\}, |x|_v = 1\}$$

where $|x|_v$ denotes the number of occurrence of v in x. For $x\sharp yw \in \mathbb{LAR}(W)$, we define $end(x\sharp yw) = w$. We define the function *next* as,

$$next(x\sharp y, w) = \begin{cases} x'\sharp x''yw & \text{iff } x\sharp y = x'wx''\sharp y \\ xy'\sharp y''w & \text{iff } x\sharp y = x\sharp y'wy'' \\ x\sharp y & \text{iff } x\sharp y = x\sharp y'w \end{cases}$$

For a sequence $\rho = w_0w_1 \ldots \in W^\omega$, we define $LAR(\rho) = x_0\sharp y_0, x_1\sharp y_1, \ldots$ where

- $x_0\sharp y_0 = x\sharp w_0$ where x consists of elements in $W \setminus \{w_0\}$ ordered according to the well ordering on W.
- for all $j > 0$, $x_j\sharp y_j = next(x_{j-1}\sharp y_{j-1}, w_j)$.

For a finite sequence ρ, the sequence $LAR(\rho)$ is finite and we denote the last LAR record in the sequence by $last(LAR(\rho))$.

One of the main applications of the LAR data structure in automata theory is to translate a Muller automaton to an equivalent Rabin automaton. The main property of the data structure, which is also crucially used to show the correctness of the translation, is stated in the following lemma.

Lemma 5.2.2 ([Tho97, Far02]) *Let ρ be an infinite sequence $w_0w_1 \ldots \in W^\omega$ and let $LAR(\rho) = x_0\sharp y_0, x_1\sharp y_1, \ldots$. Then $Inf(\rho) = F$ with $|F| = k$ iff the following conditions hold:*

- *for only finitely many j we have $|y_j| > k$ (and hence $|x_j| \leq |W| - k$).*
- *for infinitely many j we have $|y_j| = k$ (and hence $|x_j| = |W| - k$) and $F = \{w \in W \mid w \text{ occurs in } y_j\}$.*

The LAR tree

Let $G = (\mathcal{G}, \{\sqsubseteq^i\}_{i \in N})$ where $\mathcal{G} = (W, E, w_0)$ is the game arena. We assume that W is well ordered and let $\mathbb{LAR}(\mathcal{G}) = \mathbb{LAR}(W)$.

Definition 5.2.3 *For an arena $\mathcal{G} = (W, E, w_0)$ the (finite) LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$ is defined as follows: $T_{\mathbb{LAR}}(\mathcal{G}) = (S, \Rightarrow, s_0)$ where*

- $S = \mathbb{LAR}(W)$.
- $s_0 = x\sharp w_0$ *where x denotes the sequence of elements of $W \setminus \{w_0\}$ according to the well ordering.*
- $\Rightarrow \subseteq S \times S$ *satisfies the condition: for all $x\sharp yw \in S$, $x\sharp yw \Rightarrow x'\sharp y'w'$ iff*
  - $w' \in wE$.
  - $x'\sharp y'w' = next(x\sharp yw, w')$
  - *consider the unique path from the root to $x'\sharp y'w$, either there is no node in this path with the same LAR or $x'\sharp y'w$ is the first node to repeat in the path.*

The tree $T_{\mathbb{LAR}}(\mathcal{G})$ is well defined since the function *next* is well defined. The fact that $T_{\mathbb{LAR}}(\mathcal{G})$ is finite can be verified by noting that along any sequence of elements of $\mathbb{LAR}$ of length $(|W|+1)!+1$, at least one element is bound to repeat by pigeonhole principle. Let $frontier(T_{\mathbb{LAR}}(\mathcal{G}))$ denote the set of all leaf nodes of $T_{\mathbb{LAR}}(\mathcal{G})$. We define a labelling function $lab : frontier(T_{\mathbb{LAR}}(\mathcal{G})) \to 2^W$ as follows.

For a node $x'\sharp y' \in frontier(T_{\mathbb{LAR}}(\mathcal{G}))$, let ϱ be the unique path from the root $x\sharp w_0$ to $x'\sharp y'$. Let ϱ' be the least suffix of ϱ such that $first(\varrho') = last(\varrho') = x'\sharp y'$. Let $l_{max} = max\{|y| \mid x\sharp y \text{ occurs in } \varrho')\}$ and let $L_\varrho = \{x\sharp y \mid |y| = l_{max}\}$. Observe that, by the property of the LAR construction $y = y'$ for all $x\sharp y,\ x'\sharp y' \in L_\varrho$. Let $Y = \{y\}$ such that $x\sharp y \in L_\varrho$ we set $lab(x'\sharp y') = Y$.

LAR tree as a finite extensive form game: It is easy to see that $T_{\mathbb{LAR}}(\mathcal{G})$ constitutes a finite extensive form game tree where for any $i \in N$ the set of i nodes of the tree $S^i = \{x\sharp yw \in S \mid w \in W^i\}$. We use ν and η to denote the strategies of players i and $\bar{\imath}$ respectively in $T_{\mathbb{LAR}}(\mathcal{G})$. For $i \in N$, let $\Omega^i(T_{\mathbb{LAR}}(\mathcal{G}))$ denote the set of all strategies of player i in $T_{\mathbb{LAR}}(\mathcal{G})$. Note that all strategies in $\Omega^i(T_{\mathbb{LAR}}(\mathcal{G}))$ are memoryless. We use ϱ to denote plays in $T_{\mathbb{LAR}}(\mathcal{G})$. Given a profile of strategies (ν, η) in $T_{\mathbb{LAR}}(\mathcal{G})$ let $\varrho_{(\nu,\eta)}$ denote the unique resulting play.

Every strategy $\nu \in \Omega^i(T_{\mathbb{LAR}}(\mathcal{G}))$ can be translated into a bounded memory strategy $\mu \in \Omega^i(\mathcal{G})$. In other words, it can be represented as a deterministic advice automaton $\mathcal{A}_\mu = (Q_\mu, \delta_\mu, o_\mu, q_0)$ with state space Q_μ the transition function $\delta_\mu : Q_\mu \times W \to Q_\mu$ and the output function $o_\mu : Q_\mu \times W \to W$.

We define the translation function $\mathfrak{f} : \Omega^i(T_{\mathbb{LAR}}(\mathcal{G})) \rightarrow \Omega^i(\mathcal{G})$ as follows.

Definition 5.2.4 *For a strategy ν, $\mathfrak{f}(\nu) = \mu$ where μ is the strategy represented by the deterministic advice automaton $\mathcal{A}_\mu = (Q_\mu, \delta_\mu, o_\mu, q_0)$ with*

- $Q_\mu = \mathbb{LAR}(\mathcal{G})$.
- $\delta_\mu(x\sharp y, w) = next(x\sharp y, w)$.
- $o_\mu(x\sharp y, w) = \nu(x\sharp y)$.
- $q_0 = x\sharp w_0$.

We say a strategy $\mu \in \Omega^i(\mathcal{G})$ is LAR implementable if μ can be represented by an advice automaton whose state space is $\mathbb{LAR}(\mathcal{G})$ and whose transition function respects the *next* function. Let $\Omega^i_{\mathbb{LAR}}(\mathcal{G})$ denote the set of all strategies in $\Omega^i(\mathcal{G})$ which is LAR implementable. The translation function $\mathfrak{f}$ is thus a map $\mathfrak{f} : \Omega^i(T_{\mathbb{LAR}}(\mathcal{G})) \rightarrow \Omega^i_{\mathbb{LAR}}(\mathcal{G})$.

We extend the translation functions to strategy profiles as follows: For a pair of strategies (μ, τ) let $\mathfrak{g}(\mu, \tau) = (\mathfrak{g}(\mu), \mathfrak{g}(\tau))$.

The following lemmas show the relationship between the game arena and the LAR tree, it also makes clear our motivation in defining the LAR tree.

Lemma 5.2.5 *For any strategy profile (ν, η) in $T_{\mathbb{LAR}}(\mathcal{G})$ if $lab(last(\varrho_{(\nu,\eta)})) = F$ then $Inf(\rho_{(\mu,\tau)}) = F$ where $(\mu, \tau) = \mathfrak{f}(\nu, \eta)$.*

Proof: Consider any profile of strategies (ν, η) in $T_{\mathbb{LAR}}(\mathcal{G})$ and suppose we have $lab(last(\varrho_{(\nu,\eta)})) = F$. Let $\varrho_{(\nu,\eta)} = x_0\sharp w_0, x_1\sharp y_1, \ldots, x_k\sharp y_k$. By construction of the LAR tree there exists $j : 0 \leq j < k$ such that $x_j\sharp y_j = x_k\sharp y_k = x\sharp y$. Let $(\mu, \tau) = \mathfrak{f}(\nu, \eta)$ and let $\rho_{(\mu,\tau)}$ be the resulting play in $\mathcal{G}$. Let the LAR sequence of this play be $LAR(\rho_{(\mu,\tau)}) = \mathfrak{x}_0\sharp w_0, \mathfrak{x}_1\sharp\mathfrak{y}_1, \ldots$. By construction of the strategies μ and τ we have for all $r : 0 \leq r \leq k$, $\mathfrak{x}_r\sharp\mathfrak{y}_r = x_r\sharp y_r$. Thus, in particular, we have $\mathfrak{x}_j\sharp\mathfrak{y}_j = x_j\sharp y_j = x\sharp y$. In other words, consider the prefix $\rho_1 = w_0 w_1 \ldots w_j$ and $\rho_2 = w_0 w_1 \ldots w_j \ldots w_k$ of $\rho_{(\mu,\tau)}$, we have $last(LAR(\rho_1)) = last(LAR(\rho_2)) = x\sharp y$. For strategy μ, the memory state at the end of the prefix ρ_1 is $last(LAR(\rho_1)) = x\sharp y$. Suppose $w_j \in W^i$, i.e. w_j is a player i node. The choice of μ on the sequence ρ_1 is dictated by the output function $o_\mu(last(LAR(\rho_1)), w_j)$ but since $last(LAR(\rho_1)) = last(LAR(\rho_2))$ (hence also $w_j = w_k$) we have $\mu(\rho_1) = \mu(\rho_2)$. Since τ is also a bounded memory strategy based on the LAR set $\mathbb{LAR}(\mathcal{G})$, we have that the play $\rho_{(\mu,\tau)}$ settles down in the

cycle $x\sharp y, x_{j+1}\sharp y_{j+1}, \ldots, x_{k-1}\sharp y_{k-1}, x\sharp y$. By definition of the labelling function lab, there exists $p : j \leq p \leq k$ such that $\{y_p\} = F$. Therefore in $LAR(\rho_{(\mu,\tau)})$ there exist infinitely many indices m such that $\mathfrak{y}_m = |F|$ and $\{\mathfrak{y}_m\} = \{y_m\} = F$. From Lemma 5.2.2 we get $Inf(\rho_{(\mu,\tau)}) = F$. □

Backward induction algorithm

The LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$ has its frontier nodes labelled with subsets of W. We use the backward induction algorithm to extend the labelling to interior nodes of the tree as well. The procedure is as follows:

Procedure 1

- Initially, all interior nodes of $T_{\mathbb{LAR}}(\mathcal{G})$ are unlabelled.
- Repeat the following steps till $lab(x_0\sharp w_0)$ is defined, i.e. the labelling function is defined on the root node.
  - Choose any node $x\sharp y$ such that $lab(x_0\sharp w_0)$ is not defined and all of whose successors are labelled.
  - if $x\sharp y \in S^i$ then let $x_1\sharp y_1$ be a successor node such that $lab(x_2\sharp y_2) \sqsubseteq^i lab(x_1\sharp y_1)$ for all other successor nodes $x_2\sharp y_2$ of $x\sharp y$. Let $lab(x\sharp y) = lab(x_1\sharp y_1)$ and $\nu(x\sharp y) = x_1\sharp y_1$.
  - if $x\sharp y \in S^{\overline{\imath}}$ then we choose a successor $x_1\sharp y_1$ such that $lab(x_2\sharp y_2) \sqsubseteq^{\overline{\imath}} lab(x_1\sharp y_1)$ for all other successor nodes $x_2\sharp y_2$ and set $lab(x\sharp y) = lab(x_1\sharp y_1)$ and $\eta(x\sharp y) = x_1\sharp y_1$.

Consider the profile of strategies (ν, η) generated by the above procedure. From proposition 2.1.6 it follows that (ν, η) constitutes an equilibrium profile in $T_{\mathbb{LAR}}(\mathcal{G})$. We show that $\mathfrak{f}(\nu, \eta)$ constitutes an equilibrium profile in the arena $\mathcal{G}$. Before presenting the proof, we find it instructive to explain how the procedure works in the case of zero-sum games. In such games, we have a set $\mathcal{F}$ which specifies the winning condition of player i and $\overline{\mathcal{F}} = \{F \mid F \notin \mathcal{F}\}$ specifies the winning condition for player $\overline{\imath}$. The preference ordering is defined in the obvious manner: for player i, all sets in $\mathcal{F}$ are equally preferred and sets in $\mathcal{F}$ is strictly more preferred than those in $\overline{\mathcal{F}}$.

The preference ordering for player $\overline{i}$ is strictly complementary. The following lemma shows that for a player i, as far as ensuring an outcome $\mathcal{F}$ in the game arena $\mathcal{G}$ is concerned it suffices to analyse strategies generated from the LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$.

Lemma 5.2.6 *Given an arena $\mathcal{G}$ along with a Muller condition $\mathcal{F}$ for player $i \in N$, if there exists a strategy $\nu \in \Omega^i(T_{\mathbb{LAR}}(\mathcal{G}))$ such that ν ensures $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$ then $\mu = \mathfrak{f}(\nu)$ ensures $\mathcal{F}$ in $\mathcal{G}$.*

Proof: Suppose not, suppose player i can ensure $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$ by ν but cannot ensure $\mathcal{F}$ in $\mathcal{G}$ using the strategy $\mu = \mathfrak{f}(\nu)$. Then there exists a play ρ in $\mathcal{G}$ conforming to μ such that it settles down to a Muller set $F' \notin \mathcal{F}$. There are two cases to consider.

The first case is when there exists $w \in F'$ such that $w \notin F$ for any $F \in \mathcal{F}$. Let j be the first index such that $\rho(j) = w$ and $\rho(j-1) \in W^{\overline{i}}$. Let ϱ be the (finite) path in $T_{\mathbb{LAR}}(\mathcal{G})$ corresponding to ρ. The index j must be greater than $|\varrho|$; otherwise ρ couldn't have been labelled $\mathcal{F}$ and hence μ couldn't have ensured $\mathcal{F}$. Let $x'\sharp y' = LAR(\rho_{j-1})$. By the construction of $T_{\mathbb{LAR}}(\mathcal{G})$ there exists a node $x'\sharp y' \in \varrho$. But this means that player $\overline{i}$ had the option of playing w at the node $x'\sharp y'$ and hence the root to be labelled with a set in $\bar{\mathcal{F}}$. But this would contradict the fact that ν_i ensure $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$.

The other case is when there exists $F \in \mathcal{F}$ such that $w \in F$ and $w \notin F'$. Let ϱ be the (finite) path in $T_{\mathbb{LAR}}(\mathcal{G})$ corresponding to ρ. Let l be the biggest index such that $\rho(l) = w$ but $l < |\varrho|$. Suppose $\rho(l-1) \in W^i$. Then for all indices $l_1, l_2, \ldots$ such that $l < l_1 < l_2 < \ldots$ and $LAR(\rho_{l_1}) = LAR(\rho_{l_2}) = \ldots = LAR(\rho_{l-1})$, player i has to play w as it is prescribed by the strategy ν, and hence in turn by the corresponding bounded memory strategy μ. But this contradicts the fact that the ρ settles down to F'.

Finally, suppose $\rho(l-1) \in W^{\overline{i}}$. Then player $\overline{i}$ has the option of playing w at $\rho(l-1)$ and at all indices $l_1, l_2, \ldots$ such that $l < l_1 < l_2 < \ldots$ and $LAR(\rho_{l_1}) = LAR(\rho_{l_2}) = \ldots = LAR(\rho_{l-1})$. Hence ν could not have ensured $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$ as the leaf node of ϱ wouldn't have been labelled with a set in $\mathcal{F}$ and hence neither the root. □

Lemma 5.2.7 *Given an arena $\mathcal{G}$ along with a Muller condition $\mathcal{F}$ for player $i \in N$, if there exists a strategy $\mu \in \Omega^i(\mathcal{G})$ such that μ ensures $\mathcal{F}$ in $\mathcal{G}$ then there exists strategy $\nu \in \Omega^i(T_{\mathbb{LAR}}(\mathcal{G}))$ such that ν ensures $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$.*

Proof: Suppose player i does not have a strategy ν to ensure $\mathcal{F}$ in $T_{\mathbb{LAR}}(\mathcal{G})$ then $T_{\mathbb{LAR}}(\mathcal{G})$ being a finite tree (and hence a finite extensive form game) it follows that players $\overline{i}$ has a strategy η to ensure $\overline{\mathcal{F}}$ in $T_{\mathbb{LAR}}(\mathcal{G})$, since finite games are determined. Then by Lemma 5.2.6, player $\overline{i}$ has a bounded memory strategy $\tau = \mathfrak{f}(\eta)$ to ensure $\overline{\mathcal{F}}$ in $\mathcal{G}$ as well. But this contradicts the assumption that players i has a strategy to ensure $\mathcal{F}$ in $\mathcal{G}$. □

Theorem 5.2.8 then follows from Lemmas 5.2.6 and 5.2.7.

Theorem 5.2.8 *Given an arena $\mathcal{G}$ and along with Muller condition $\mathcal{F}$ for player $i \in N$ there exists a strategy μ for player i to ensure $\mathcal{F}$ in $\mathcal{G}$ iff player i has a strategy in $T_{\mathbb{LAR}}(\mathcal{G})$ to ensure $\mathcal{F}$.*

Theorem 5.2.9 *Every generalised Muller game has a Nash equilibrium.*

Proof: Let $G = (\mathcal{G}, \{\sqsubseteq^i\}_{i \in N})$ be a generalised Muller game. Consider the strategy profile (ν, μ) generated by the backward induction procedure on the LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$. We show that the strategy tuple $(\mu, \tau) = \mathfrak{f}(\nu, \eta)$ constitutes an equilibrium profile in $\mathcal{G}$. The proof is similar to that of Theorem 5.2.8: we show that for $i \in N$, player i has an incentive to deviate from μ in $\mathcal{G}$ iff she has an incentive to deviate from μ in the LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$.

Suppose player i deviates to strategy μ'. Let ρ be the run corresponding to (μ', τ) with $Inf(\rho) = F'$ and $X = \{w \in F' \mid w \notin F\}$. Suppose $X \neq \emptyset$, let j be the first index such that $\rho(j) \in X$ and let $LAR(\rho(j-1)) = x'\sharp y'$. By the construction, there exists a node $x'\sharp y'$ in the LAR tree. Since player $\overline{i}$ plays according to the LAR strategy τ derived from the strategy η, it can be seen that the node $x'\sharp y'$ is reachable in the LAR tree by player i's deviation. But then we have that player i has a option of playing w at $x'\sharp y'$ and since the LAR is the same, she can choose a path in $T_{\mathbb{LAR}}(\mathcal{G})$ which is labelled with F'.

Let $Y = \{w \in F \mid w \notin F'\}$. If $Y \neq \emptyset$, then let j be the last index such that $\rho(j) \in Y$ and let $LAR(\rho(j)) = x'\sharp y'$. As a result of the deviation, player i ensures that elements in Y are visited only finitely many times. By construction we have that $x'\sharp y'$ is present in the LAR tree. As earlier it can be seen that $x'\sharp y'$ is reachable by the deviation of player i. From $\rho(j)$, player i ensures that elements of the set X are never visited. But since we have the same LAR and since player $\overline{i}$ uses the LAR strategy τ derived from η this means that from $x'\sharp y'$, player i can play in such a way that the resulting path is labelled with F'.

□

Note that since the translation function ƒ is effective, we also get that the equilibrium profile for any generalised Muller game can be synthesized in finite memory strategies.

Complexity: Let the number of vertices in the arena $\mathcal{G}$ be m. The size of the LAR memory is $\mathcal{O}(m!)$. As there are $\mathcal{O}(m^{m!})$ paths in the LAR tree $T_{\mathbb{LAR}}(\mathcal{G})$ and the backward induction procedure runs in time linear in the size of the LAR tree, the running time is $\mathcal{O}(m^{m!})$.

5.3 Partial strategies and best response computation

In the previous section we looked at equilibrium computation in non-zero sum infinite games with respect to functional strategies which depict complete plans. A natural question would be to ask whether strategy specifications help in the analysis of such games. In order to analyse specifications in terms of solution concepts, we need to first define on what basis specifications can be compared with each other. In the case of complete strategies, given a strategy τ of player $\bar{i}$, comparison between two strategies μ and μ' of player i was defined in terms of the unique outcome which is achieved. However, this definition is not suitable in the case of strategy specifications since we are dealing with a set of strategies. Thus in the context of strategy specifications basic notions like strategy comparison and best response need to be revisited.

Given a game arena $G = (\mathcal{G}, \mathcal{E})$ and a strategy specification π for player $\bar{i}$, we can have different notions as to when a specification for player i is "better" than another.

- $Better_1(\sigma, \sigma')$: For some $F \in 2^R$, **if** ($\exists \mu'$ with $\mu' \models_i \sigma'$ such that $\forall \tau$ with $\tau \models_{\bar{i}} \pi$, $\rho^{\tau}_{\mu'}$ is winning with respect to $\mathcal{E}^i_F$) **then** ($\exists \mu$ with $\mu \models_i \sigma$ such that $\forall \tau$ with $\tau \models_{\bar{i}} \pi$, ρ^{τ}_{μ} is winning with respect to $\mathcal{E}^i_F$).

  The predicate $Better_1(\sigma, \sigma')$ says that, for some (binary) outcome F, if there is a strategy conforming to the specification σ' which ensures winning $\mathcal{E}^i_F$ then there also exists a strategy conforming to σ which ensures winning $\mathcal{E}^i_F$ as well.

- *Better*$_2(\sigma, \sigma')$: For some $F \in 2^R$, **if** (for all strategies μ' with $\mu' \models_i \sigma'$ and $\forall \tau$ with $\tau \models_{\overline{i}} \pi$, $\rho^\tau_{\mu'}$ is winning with respect to $\mathcal{E}^i_F$) **then** ($\forall \mu$ with $\mu \models_i \sigma$ and $\forall \tau$ with $\tau \models_{\overline{i}} \pi$, ρ^τ_{μ} is winning with respect to $\mathcal{E}^i_F$).

  This notion is best understood contrapositively: for some (binary) outcome F, whenever there is a strategy conforming to σ which is not winning for $\mathcal{E}^i_F$, there also exists a strategy conforming to σ' which is not winning for $\mathcal{E}^i_F$. This can be thought of as a soundness condition. A risk averse player might prefer this notion of comparison.

The two above are just a few of the various possibilities for strategy comparison, whose actual choice might depend on the kind of application in mind. Having chosen the appropriate notion, we say that σ is the best response to π, if for all σ', we have σ is better (according to that notion) than σ'. A strategy pair (σ, π) is said to be in equilibrium if σ is the best response to π and π is the best response to σ.

To algorithmically compare strategies, we first need to be able to decide the following questions. Let σ and π be strategy specifications for player i and player $\overline{i}$ and $\mathcal{E}^i_F$ a binary evaluation automaton for player i.

- Does player i have a strategy conforming to σ which is winning for i with respect to $\mathcal{E}^i_F$, against all strategies of player $\overline{i}$ which conforms to π (abbreviated as $\exists \sigma, \forall \pi : \mathcal{E}^i_F$)?

- Is it the case that for all strategies of player i conforming to σ, as long as player $\overline{i}$ is playing a strategy conforming to π, the result will be a valid play which is winning for i with respect to $\mathcal{E}^i_F$ (abbreviated as $\forall \sigma, \forall \pi : \mathcal{E}^i_F$)?

We call this the verification question. The synthesis question is given π and $\mathcal{E}^i_F$ to construct a deterministic advice automaton $\mathcal{A}$ for player i such that $\mathcal{A}, \forall \pi : \mathcal{E}^i_F$ holds.

Once we can show that the verification question is decidable and synthesis possible, the game theoretic questions of interest include: For a game $G = (\mathcal{G}, \mathcal{E})$,

- Given strategy specifications σ and π, check if σ is a best response to π.

- Given a strategy specification profile $\langle \sigma, \pi \rangle$, check if it is a Nash equilibrium.

- Given a strategy specification π for $\overline{i}$, synthesize a deterministic advice automaton $\mathcal{A}$ for player i such that $\mathcal{A}$ is the best response to π.

Definition 5.3.1 *Let $\mathcal{G} = (W, \rightarrow, w_0, \lambda)$ be an arena and $\mathcal{A}_\sigma = (Q, \delta, o, I)$ be the advice automaton corresponding to a strategy specification σ of player i. The restriction of $\mathcal{G}$ with respect to $\mathcal{A}_\sigma$ is the structure $\mathcal{G} \upharpoonright \mathcal{A}_\sigma = (W_\sigma, \rightarrow_\sigma, w_0^\sigma, \lambda_\sigma)$ where*

- $W_\sigma = W \times 2^Q$ *is the set of game positions.*
- $(w, X) \xrightarrow{a}_\sigma (w', Y)$ *iff* $Y = \{q' \mid \exists q \in X \text{ with } q' \in \delta(q, w, a)\}$.
- $w_0^\sigma = (w_0, I_\sigma)$.
- $\lambda_\sigma(w, X) = \lambda(w)$.

It is easy to see that the arena $\mathcal{A}_\sigma$ is deterministic and satisfies the property:

(R1) for all $(w, X) \in W_\sigma$, for all $a \in \Sigma$ such that $w \xrightarrow{a} w'$, there exists a unique Y such that $(w, X) \xrightarrow{a}_\sigma (w', Y)$.

Every strategy $\nu \in \Omega^i(\mathcal{G} \upharpoonright \mathcal{A}_\sigma)$ is also a strategy in $\Omega^i(\mathcal{G})$ where the additional memory required is 2^Q. Given a strategy $\nu = (S_\nu, \Rightarrow_\nu, s_0^\nu, \widehat{\lambda}_\nu)$ let $st(\nu)$ be the strategy tree obtained by simply projecting out the 2^Q component from the game positions of ν. Due to property (R1) this defines a valid strategy in $\Omega^i(\mathcal{G})$. Lemma 5.3.2 follows from the definition of the restriction operation.

Lemma 5.3.2 *For all $\nu \in \Omega^i(\mathcal{G} \upharpoonright \mathcal{A}_\sigma)$, $st(\nu) \in Lang(\mathcal{A}_\sigma)$.*

Lemma 5.3.3 *For all $\mu \in \Omega^i(\mathcal{G})$ such that $\mu \in Lang(\mathcal{A}_\sigma)$, there exists $\nu \in \Omega^i(\mathcal{G} \upharpoonright \mathcal{A}_\sigma)$ such that $st(\nu) = \mu$.*

Proof: Consider any $\mu \in \Omega^i(\mathcal{G})$ such that $\mu \in Lang(\mathcal{A}_\sigma)$. Let $T = (S_\mu, \Rightarrow_\mu, s_0^\mu, \widehat{\lambda}_\mu, l)$ be Q labelled tree accepted by $\mathcal{A}_\sigma$. We define the strategy $\nu = (S_\nu, \Rightarrow_\nu, s_0^\nu, \widehat{\lambda}_\nu)$ inductively. Let ν_0 be the tree containing the single node (w_0, I). The construction maintains the following invariant property:

(Inv1) for all $t \in \nu$, where $t[1] = s$ and $last(t) = (w, X)$, if $l(s) = q$ then $q \in X$.

Since $l(s_0) \in I$, for the tree ν_0, property 1 is satisfied. Assume inductively we have constructed the tree $\nu_k = (S_\nu, \Rightarrow_\nu, s_0^\nu, \widehat{\lambda}_\nu)$. Pick any node $t \in S_\nu$ where $t[1] = s$ and $last(t) = (w, X)$. We have the following two cases:

1. If $\widehat{\lambda}_\nu(t) = i$ then let $a = o_\sigma(l(s), w)$. By definition, there exists a unique outgoing edge in μ such that $s \stackrel{a}{\Rightarrow} s'$, let $w' = last(s')$. By property 1, we get $l(s) \in X$. By construction of $\mathcal{G} \upharpoonright \mathcal{A}_\sigma$, there exists a unique node (w', Y) such

that $(w, X) \xrightarrow{a}_\sigma (w', Y)$. Since T is an accepting run, $l(s') \in \delta(l(s), w, a)$ and by definition of $\mathcal{G} \restriction \mathcal{A}_\sigma$ we have $l(s') \in Y$. Thus the invariant property can be maintained by extending the tree with the node $t' = t \cdot (w', Y)$.

Define the tree $\nu_{k+1} = (S_{\nu_{k+1}}, \Rightarrow_{\nu_{k+1}}, s_0^{\nu_{k+1}}, \widehat{\lambda}_{\nu_{k+1}})$ where $S_{\nu_{k+1}} = S_{\nu_k} \cup \{t'\}$ and $\Rightarrow_{\nu_{k+1}} = \Rightarrow_{\nu_k} \cup \{t \overset{a}{\Rightarrow} t'\}$.

2. If $\widehat{\lambda}_\nu(t) = \overline{i}$ let $\{s_1, \ldots, s_m\}$ be the set of all nodes in μ such that $s \overset{a_j}{\Rightarrow} s_j$ for all $j : 1 \leq j \leq m$. Let $last(s_j) = w_j$ and $q_j = l(s_j)$. By construction of $\mathcal{G} \restriction \mathcal{A}_\sigma$, for all j we have $(w, X) \xrightarrow{a}_\sigma (w_j, Y_j)$ and $q_j \in Y_j$. Let $t_j = t \cdot (w_j, Y_j)$.

   Define the tree $\nu_{k+1} = (S_{\nu_{k+1}}, \Rightarrow_{\nu_{k+1}}, s_0^{\nu_{k+1}}, \widehat{\lambda}_{\nu_{k+1}})$ where $S_{\nu_{k+1}} = S_{\nu_k} \cup \{t_1, \ldots, t_m\}$ and $\Rightarrow_{\nu_{k+1}} = \Rightarrow_{\nu_k} \cup \{t \overset{a_1}{\Rightarrow} t_1, \ldots, t \overset{a_m}{\Rightarrow} t_m\}$.

The strategy $\nu = (S_\nu, \Rightarrow_\nu, s_0^\nu, \widehat{\lambda}_\nu)$ defined by $S_\nu = \bigcup_{k \geq 0} S_{\nu_k}$ and $\Rightarrow_\nu = \bigcup_{k \geq 0} \Rightarrow_{\nu_k}$. From the construction it follows that $\nu \in \Omega^i(\mathcal{G} \restriction \mathcal{A}_\sigma)$ and it is also easy to see that $st(\nu) = \mu$. □

Lemma 5.3.4 *For every strategy $\mu \in \Omega^i(\mathcal{G})$, for all $i \in N$ and for all $\sigma \in Strat^i(P^i)$, there exists $\mu' \in \Omega^i(\mathcal{G} \restriction \mathcal{A}_\sigma)$ such that $st(\mu') = \mu$ iff $\mu \models_i \sigma$.*

Proof: Follows from Lemmas 5.3.2, 5.3.3 and 3.1.7. □

In other words, Lemma 5.3.4 states that strategies of player i in the restricted arena $\mathcal{G} \restriction \mathcal{A}_\sigma$ are precisely those strategies of i in $\mathcal{G}$ which conform to σ. The restriction operation can be applied iteratively. For instance, given advice automata $\mathcal{A}_\sigma$ and $\mathcal{A}_\pi$ of players i and $\overline{i}$ respectively, the structure $(\mathcal{G} \restriction \mathcal{A}_\sigma) \restriction \mathcal{A}_\pi$ consists of all paths which conform to the specifications σ and π. It is also easy to check that the order of restriction is irrelevant. That is $(\mathcal{G} \restriction \mathcal{A}_\sigma) \restriction \mathcal{A}_\pi = (\mathcal{G} \restriction \mathcal{A}_\pi) \restriction \mathcal{A}_\sigma$

Theorem 5.3.5 *Given a game $G = (\mathcal{G}, \mathcal{E})$ and a strategy specification π for player $\overline{i}$,*

1. *The verification problem of checking whether for a player i a strategy specification σ and a binary evaluation automaton $\mathcal{E}_F^i$, checking whether $\exists\sigma, \forall\pi : \mathcal{E}_F^i$ or $\forall\sigma, \forall\pi : \mathcal{E}_F^i$ holds in $\mathcal{G}$ is decidable.*

2. *For a binary evaluation automaton $\mathcal{E}_F^i$, it is possible to synthesize (when one exists), a deterministic advice automaton $\mathcal{A}_i$ such that $\mathcal{A}_i, \forall\pi : \mathcal{E}_F^i$ holds.*

Proof: The assertion $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ holds in the arena $\mathcal{G}$ iff there exists strategy μ for player i which conforms to σ such that for all strategies τ of player $\overline{i}$ conforming to π, the resulting play $\rho_{(\mu,\tau)}$ is "winning" for player i with respect to the win-loss condition given by $\mathcal{E}^i_F$. We make use of the restriction operation given in definition 5.3.1 to decide the verification question. Let the advice automata corresponding to σ and π be $\mathcal{A}_\sigma$ and $\mathcal{A}_\pi$ respectively. Consider the arena $\mathcal{G} \restriction \mathcal{A}_\pi$; by Lemma 5.3.4 strategies of player $\overline{i}$ in the restricted arena are precisely the strategies which conform to π in $\mathcal{G}$. Thus to check if $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ holds it suffices to check if there exists a strategy for player i conforming to σ in $\mathcal{G} \restriction \mathcal{A}_\pi$ which is winning for the objective given by $\mathcal{E}^i_F$.

We construct a nondeterministic tree automaton $\mathcal{T}$ which checks this property. Intuitively, the automaton works as follows: it simulates both $\mathcal{A}_\sigma$ and $\mathcal{E}^i_F$ and runs on $T_{\mathcal{G} \restriction \mathcal{A}_\pi}$. Thus the states of the tree automaton are tuples of the form (q, r) and the initial state is (q_0, r_0) where $q_0 \in I$. At any position s of $T_{\mathcal{G} \restriction \mathcal{A}_\pi}$ where the state of the automaton is (q, r), the automaton proceeds as follows:

- if s is a player i game position then let a be the action dictated by the output function o of $\mathcal{A}_\sigma$ on state q and position s. The automaton guesses a new state $q' \in \delta(q, s, a)$ and proceeds down the a edge on the state q'. Formally, this means that on all outgoing edges of s labelled by $b \neq a$ the automaton enters a default accept state and stays in this accept state.

- if s is a player $\overline{i}$ game position then let $\{b_1, \ldots, b_k\}$ be the outgoing edges at s. For each action b_j player i guesses a state $q_j \in \delta(q, s, a)$ and branches on all the outgoing edges.

In other words, the automaton $\mathcal{T}$ guesses a strategy μ of player i in $\mathcal{G} \restriction \mathcal{A}_\pi$ which conforms to σ. $\mathcal{T}$ accepts this strategy if all paths of μ are accepted by the Muller automaton $\mathcal{E}^i_F$.

Formally, the tree automaton is given by: $\mathcal{T} = (\mathbb{Q}, \mathbb{R}, \mathbb{I})$ where $\mathbb{Q} = (Q_\sigma \times R)$ and $\mathbb{I} = I_\sigma \times r_0$. For $\mathcal{T}$ in a state q, reading node t, $\mathbb{R}(q, t) = \langle (q_1, a_1), \ldots, (q_m, a_m) \rangle$ means that the automaton will branch out: on the a_1 successor it goes into state q_1, a_2 successor it goes to state q_2 and so on. The transition relation is defined as follows: for a node $t \in S_\pi$, let $\{a_1, \ldots, a_m\}$ be the outgoing edges.

- If $\widehat{\lambda}_\pi(t) = i$ then $\mathbb{R}((q, r), t) = \{\langle (accept, a_1), \ldots, ((q_j, r_j), a_j), \ldots, (accept, a_m) \rangle \mid o_\sigma(q_j, t) = a_j, q_j \in \delta_\sigma(q, t, a_j) \text{ and } r_j = \Delta(r, t, a_j)\}$.

- If $\widehat{\lambda}_\pi(t) = \overline{i}$ then $\mathbb{R}((q,r),t) = \{\langle((q_1,r_1),a_1),\ldots,((q_j,r_j),a_j),\ldots,((q_m,r_m),a_m)\rangle \mid q_j \in \delta_\sigma(q,t,a_j)$ and $r_j = \Delta(r,t,a_j)$ for all $j : 0 \leq j \leq m\}$.

The tree automaton $\mathcal{T}$ accepts a tree iff for all paths either the Muller condition specified by the win-loss automaton $\mathcal{E}_F^i$ is satisfied or the state *accept* occurs infinitely often.

To check if $\forall\sigma, \forall\pi : \mathcal{E}_F^i$ holds, it suffices to check if all plays in $(\mathcal{G} \upharpoonright \mathcal{A}_\pi) \upharpoonright \mathcal{A}_\sigma$ are winning for i with respect to $\mathcal{E}_F^i$. This can be done easily.
(2) We want to synthesize a bounded memory strategy μ for player i such that for all strategies τ of player $\overline{i}$ conforming to π, the resulting play $\rho_{(\mu,\tau)}$ is winning for player i with respect to $\mathcal{E}_F^i$. By the observation made in the previous part, we need to synthesize a bounded memory winning strategy for player i (if it exists) in the restricted game $\mathcal{G} \upharpoonright \mathcal{A}_\pi$. Consider the game $(\mathcal{G} \upharpoonright \mathcal{A}_\pi, \mathcal{E}_F^i)$, this constitutes a classical win-loss Muller game where the arena is $\mathcal{G} \upharpoonright \mathcal{A}_\pi$ and the winning condition of player i is given by $\mathcal{E}_F^i$. We know that if player i has a winning strategy in the game $(\mathcal{G} \upharpoonright \mathcal{A}_\pi, \mathcal{E}_F^i)$ then i has a bounded memory winning strategy μ which can be synthesized effectively. The advice automaton $\mathcal{A}_i$ is taken to be the automaton representing the bounded memory winning strategy μ. □

Theorem 5.3.6 *Given a game $G = (\mathcal{G}, \mathcal{E})$ and a strategy specification π for player $\overline{i}$,*

1. *For a specification σ for player i, checking if σ is the best response to π is decidable.*
2. *It is possible to synthesize a deterministic advice automaton $\mathcal{A}_i$ such that $\mathcal{A}_i$ is the best response to π.*

Proof: (1): Given σ and π to check if σ is the best response to π, we use the tree automaton construction in Theorem 5.3.5 with a slight modification. We enumerate the sets $F \in \mathcal{F}$ in such a way that those higher in $\lhd^i$ appear earlier in the enumeration. For each F, we construct a tree automaton as in Theorem 5.3.5, the only difference being that the guesses made by $\mathcal{T}$ at player i game positions are not restricted by σ. $\mathcal{T}$ runs $\mathcal{E}_F^i$ in parallel to check if player i can ensure F for all choices of $\overline{i}$ which conform to π. Since the evaluation automaton is "complete", the play eventually settles down in one of the sets $F' \in \mathcal{F}$. Therefore, as we try elements of $\mathcal{F}$ in order, the tree automaton succeeds for some $\mathcal{E}_{F'}^i$. This gives us the "best" outcome which player i can guarantee. We then use the verification procedure given in theorem 5.3.5 to check if $\exists\sigma, \forall\pi : \mathcal{E}_{F'}^i$ holds in $\mathcal{G}$.

This also implies that given a strategy profile (presented as advice automata), we can verify whether the profile constitutes a Nash equilibrium.

For (2) we enumerate $\mathcal{F}$ and find the "best" outcome F that can be achieved. Using the synthesis procedure given in theorem 5.3.5, we then synthesize an advice automaton for F. □

Chapter 6

Games with imperfect information

So far in this thesis we have looked at games of perfect information. Games which capture more realistic situations of social interaction are ones where players do not have complete information on the past moves of other players. These are games of imperfect information. The typical way of modelling imperfect information in games is in terms of information partitions for players. In this view, each player i is associated with an equivalence relation $\sim^i$ over the set of game positions. For two game positions w and w' if $w \sim^i w'$ it means that player i cannot distinguish whether the current game position is w or w'. The equivalence relation is part of the game definition and it is assumed to be presented along with the game structure.

Now consider the following n player game with players $0, \ldots, n-1$. Each player has a local arena (graph). The global arena is constructed by taking the product of the local arenas such that it satisfies the condition: player 0 has access to the global game positions whereas players 1 to n can view only their local graph structures. For $i \in \{1, \ldots, n\}$, the view of player i is the history of the play restricted to i's local game structure. Imperfect information arises from the fact that player i is not aware of the exact global state but only his local component of the global state. The information sets of player i constitute the global game positions where his views are the same. A local strategy of player i dictates his choice based on his view. The objective can be taken to be a regular win-loss condition. We can now ask the following verification question:

- does there exist a tuple of winning local strategies for players $\{1, \ldots, n\}$ in $\mathcal{G}$?

It follows from the result by Peterson and Reif [PR79] that this question is undecidable. Thus the verification question is undecidable for the general class of multi-player games with imperfect information. The global game arena has its structure

derived from the local game graph. However, there is no information passed between players about their local game structure.

A closely related question is the synthesis question which asks whether it is possible to synthesize local winning strategies when they exist. This has been extensively studied in the control theory literature where the synthesis of distributed control can be modelled as a game where n players playing against a global environment. Pnueli and Rosner show that even for a two site distributed architecture where there is no possible communication between the sites, the synthesis question is undecidable [PR90]. There has been various work in this context which investigates conditions required to attain decidability of synthesis (see [Mad01], [KV01] for an overview). [MW03] proposes a model for distributed games in order to formalize and solve distributed synthesis problems.

In this chapter we propose a model for imperfect information games, where the information partitions are generated explicitly by players' behaviour. Communication between players is part of the game model and thus imperfect information depends on the exact mechanism of communication adopted by players. We show that in the case when players communicate by means of public announcements, the verification question is indeed decidable. This also suggests that the real problem lies not in the fact that there is imperfect information but rather in the way it arises. If uncertainty is introduced through some structural means then it may be possible to resolve it using communication.

6.1 The game model

Since we are looking at games of imperfect information, we deal with multiple players explicitly. Let $N = \{1, \ldots, n\}$ be the set of players. We want to make communication between players explicit in the model. For this purpose we associate with each player $i \in N$ a finite set Γ^i which represents the set of symbols which player i can employ for communication. Let $\widetilde{\Gamma} = \Gamma^1 \times \cdots \times \Gamma^n$.

6.1.1 Game arena

Local arena: For a player $i \in N$, the local game arena for player i is given by $\mathcal{G}^i = (W^i, \rightarrow_i, w_0^i, \chi^i)$ where

- W^i is a finite set of local game positions.

- w_0^i is the initial game position.
- $\chi^i : W^i \to \Gamma^i$ associates with each local game position of player i an element of Γ^i.
- $\to_i : W^i \times \widetilde{\Gamma} \to 2^{W^i}$ is the move function which satisfies the following condition: for all $w^i, v^i \in W^i$, if $w^i \xrightarrow{\gamma}_i v^i$ then $\gamma(i) = \chi^i(w^i)$.

The local game arena dictates the rules of the game for each player i. For each local game position w^i, the function χ^i specifies what player i communicates with the other players. The transition function takes into account the current game position and the communication received from other players to specify the set of possible moves enabled for player i. Note that communication in this model is by means of public announcements since for any state w^i, the value of $\chi^i(w^i)$ is communicated to all players. A game structure $\mathbb{G}$ is defined in terms of a set of local game arenas for each player, $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$.

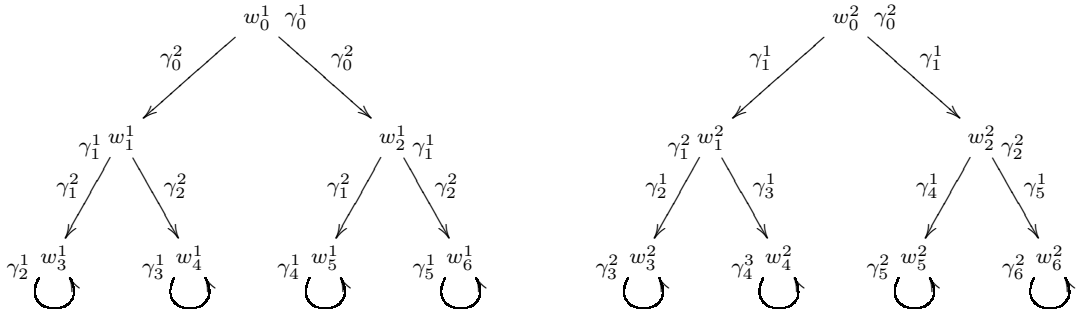
Global arena: Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, the resulting global game arena $\mathcal{G} = (W, \to, \mathbf{w}_0)$ is constructed as follows: the set of global game positions $W = W^1 \times \cdots \times W^n$ and $\mathbf{w}_0 = (w_0^1, \ldots, w_0^n)$. We define the function $\chi : W \to \widetilde{\Gamma}$ as $\chi(\mathbf{w}) = (\chi^1(w^1), \ldots, \chi^n(w^n))$ which associates with each global state, the announcements of players. The move relation $\to \subseteq W \times W$ satisfies the property: for all $\mathbf{w}, \mathbf{v} \in W$ we have $\mathbf{w} \to \mathbf{v}$ iff

- $\forall i \in \mathit{enabled}(\mathbf{w})$, $w^i \xrightarrow{\gamma}_i v^i$ or $w^i = v^i$, where $\gamma = \chi(\mathbf{w})$.
- $\forall i \in N \setminus \mathit{enabled}(\mathbf{w})$, $v^i = w^i$.

where $\mathit{enabled}(\mathbf{w}) = \{i \in N \mid \exists v^i \in W^i \text{ with } w^i \xrightarrow{\gamma}_i v^i \text{ where } \gamma = \chi(\mathbf{w})\}$.

Note that according to the global transition relation, for a player i at a global state $\mathbf{w}$, even if a move of player i is enabled at $\mathbf{w}$ the player has the option of remaining in the same state and choosing not to move.

Example 6.1.1 Let the players be $N = \{1, 2\}$ and the communication alphabets be $\Gamma^1 = \{\gamma_0^1, \ldots, \gamma_5^1\}$ and $\Gamma^2 = \{\gamma_0^2, \ldots, \gamma_6^2\}$. Consider the local game arenas $\mathcal{G}^1$ of player 1 given in Figure 6.1(a). The nodes of the graph corresponds to the local game positions, the announcements made by player 1 at each local state is marked along with the local states. For instance, $\chi^1(w_0^1) = \gamma_0^1$, $\chi^1(w_1^1) = \chi^1(w_2^1) = \gamma_1^1$ and so on. The self loop on states without any announcement annotation means that irrespective of



(a) $\mathcal{G}^1$ (b) $\mathcal{G}^2$

Figure 6.1: Local game arenas

the announcement made by the other player, the local state remains the same. The local arena $\mathcal{G}^2$ for player 2 is given in Figure 6.1(b). The derived global game graph is shown in Figure 6.2. We have not shown the possibilities of players delaying moves in the global arena. In the global game graph, players 1 and 2 alternate moves till the game reaches one of the global sink nodes $\{(w_3^1, w_3^2), (w_4^1, w_4^2), (w_5^1, w_5^2), (w_6^1, w_6^2)\}$. Player 2 cannot distinguish between the global states (w_1^1, w_0^2) and (w_2^1, w_0^2) since $view^2((w_0^1, w_0^2)(w_1^1, w_0^2)) = view^2((w_0^1, w_0^2)(w_2^1, w_0^2))$.

The model does allow players to resolve imperfect information as the play progresses. For instance at the global state (w_3^1, w_3^2) player 2 knows that the play passed through the position (w_1^1, w_0^2) and not through (w_2^1, w_0^2). □

Since the global game arena is derived from the local arenas it is possible that there exist global game positions where moves of none of the players are enabled. In other words, these are game positions where no progress can be made any further. For convenience, we think of such terminal game positions as sink nodes with a self loop. Thus a play in $\mathcal{G}$ is an infinite path $\rho = \mathbf{w}_0\mathbf{w}_1 \ldots$ such that for all $j > 0$, we have $\mathbf{w}_{j-1} \rightarrow \mathbf{w}_j$. We denote the set of all plays in $\mathcal{G}$ by $Plays(\mathcal{G})$. We also use the notation $\mathfrak{P}(\mathcal{G})$ to denote the set of all finite partial plays in $\mathcal{G}$. For a partial play ϱ, we let $enabled(\varrho) = enabled(last(\varrho))$.

The (infinite) extensive form game tree $T_{\mathcal{G}}$ associated with $\mathcal{G}$ is obtained by the tree unfolding of $\mathcal{G}$. In the tree unfolding, in addition to keeping track of the sequence of game positions, (according to Definition 2.3.2) we also keep track of the sequence of announcements made by players. Formally the tree unfolding is defined as follows:

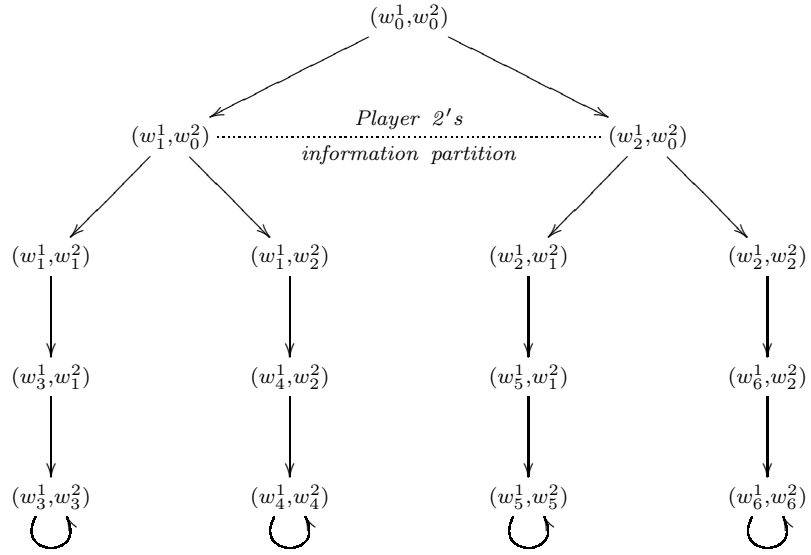


Figure 6.2: Global game arena

Definition 6.1.2 *Let $\mathcal{G} = (W, \rightarrow, \mathbf{w}_0)$ be a global game arena. The tree unfolding of $\mathcal{G}$ is the structure $T_{\mathcal{G}} = (S, \Rightarrow, s_0)$ where $S \subseteq (W \times \widetilde{\Gamma})^+$ and $\Rightarrow \subseteq S \times S$ are the least sets satisfying:*

- $(\mathbf{w}_0, \chi(\mathbf{w}_0)) \in S$.
- *If $s = (\mathbf{w}_0, \gamma_0) \ldots (\mathbf{w}_k, \gamma_k) \in S$ and $\mathbf{w}_k \rightarrow \mathbf{w}'$ then $s' = (\mathbf{w}_0, \gamma_0) \ldots (\mathbf{w}_k, \gamma_k)(\mathbf{w}', \chi(\mathbf{w}')) \in S$ and $s \Rightarrow s'$.*

6.1.2 Strategies

Views of players: For $i \in N$ and $\mathbf{w} \in W$, player i's view of $\mathbf{w}$ is defined as $view^i(\mathbf{w}) = (w^i, \chi(\mathbf{w}))$. For a sequence $\rho : \mathbf{w}_0\mathbf{w}_1 \ldots$, we define player i's view of ρ as $view^i(\rho) = view^i(\mathbf{w}_0)view^i(\mathbf{w}_1) \ldots$. Let $Plays^i(\mathcal{G}) = \{view^i(\rho) \mid \rho \in Plays(\mathcal{G})\}$. Note that a sequence in $Plays^i(\mathcal{G})$ need not necessarily be a path in $\mathcal{G}^i$.

Strategies of players: A strategy for player i, is a function $\mu^i : W^* \rightarrow W^i$ which satisfies the following conditions:

($\mathcal{S}1$) $\mu^i(\epsilon) = w_0^i$.

($\mathcal{S}2$) for all finite partial plays $\varrho \in \mathfrak{P}(\mathcal{G})$, if $\mu^i(\varrho) = w^i$ then there exists $\mathbf{v}$ such that $last(\varrho) \rightarrow \mathbf{v}$ and $v^i = w^i$. This says that the strategy must choose only moves which are enabled.

($\mathcal{S}3$) For all $\varrho, \varrho' \in \mathfrak{P}(\mathcal{G})$, if $view^i(\varrho) = view^i(\varrho')$ then $\mu^i(\varrho) = \mu^i(\varrho')$. This says that the strategy needs to respect the information partition.

We say that a play $\rho : \mathbf{w}_0\mathbf{w}_1 \ldots$ is consistent with strategy μ^i if $\forall j > 0$, $w^i_j = \mu^i(view^i(\mathbf{w}_0 \ldots \mathbf{w}_{j-1}))$.

As we saw earlier (in Section 2.1.2) with a strategy μ^i of player i, we can associate a strategy tree. Rather than viewing the strategy tree as a subtree of $T_{\mathcal{G}}$, in this chapter we represent it in terms of a labelling function on nodes of $T_{\mathcal{G}}$.

Definition 6.1.3 *Given a game tree $T_{\mathcal{G}}$ and a strategy μ^i of player i, the strategy tree $t_{\mu^i} = (T_{\mathcal{G}}, \mathfrak{l})$ where $\mathfrak{l} : S \to W^i$ is defined as $\mathfrak{l}(s) = \mu^i(s)$.*

It is easy to see that a strategy profile $\overline{\mu} = (\mu^1, \ldots, \mu^n)$ generates a unique path in $\mathcal{G}$, we denote this by ρ_μ.

Joint and distributed strategies: The definition of view can be extended to a subset of players in the natural manner. For $\mathcal{C} = \{i_1, \ldots, i_k\} \subseteq N$ and a position $\mathbf{w} \in W$, we have $view^{\mathcal{C}}(\mathbf{w}) = (w^{\mathcal{C}}, \chi(\mathbf{w}))$ where $w^{\mathcal{C}} = (w^{i_1}, \ldots, w^{i_k})$. For a play $\rho : \mathbf{w}_0\mathbf{w}_1 \ldots$ we have $view^{\mathcal{C}}(\rho) = view^{\mathcal{C}}(\mathbf{w}_0)view^{\mathcal{C}}(\mathbf{w}_1) \ldots$. Let $Plays^{\mathcal{C}}(\mathcal{G}) = \{view^{\mathcal{C}}(\rho) \mid \rho \in Plays(\mathcal{G})\}$.

For a subset of players $\mathcal{C} = \{i_1, \ldots, i_k\} \subseteq N$, a joint $\mathcal{C}$-strategy is a function $\tau^{\mathcal{C}} : W^* \to W^{\mathcal{C}}$ which satisfies the following conditions:

- $\tau^{\mathcal{C}}(\epsilon) = (w_0^{i_1}, \ldots, w_0^{i_k})$.
- $\forall \varrho \in \mathfrak{P}(\mathcal{G})$ if $\tau^{\mathcal{C}}(\varrho) = w^{\mathcal{C}}$ then there exists $\mathbf{v}$ such that $last(\varrho) \to \mathbf{v}$ and $v^{\mathcal{C}} = w^{\mathcal{C}}$.
- For all $\varrho, \varrho' \in \mathfrak{P}(\mathcal{G})$, if $view^{\mathcal{C}}(\varrho) = view^{\mathcal{C}}(\varrho')$ then $\tau^{\mathcal{C}}(\varrho) = \tau^{\mathcal{C}}(\varrho')$.

A *distributed $\mathcal{C}$-strategy* is a tuple of strategies $\mu^{\mathcal{C}} = (\mu^{i_1}, \ldots, \mu^{i_k})$. A distributed $\mathcal{C}$-strategy μ defines a joint $\mathcal{C}$-strategy as follows: for all partial plays $\varrho \in \mathfrak{P}(\mathcal{G})$, $\tau^{\mathcal{C}}(\varrho) = (\mu^{i_1}(\varrho), \ldots, \mu^{i_k}(\varrho))$.

When $\mathcal{C}$ consists of a single player i, a strategy μ^i for player i constitutes a joint $\mathcal{C}$-strategy as well as a distributed $\mathcal{C}$-strategy.

6.1.3 Objectives of players

The game arena defines the rules of the game, we first look at the situation where with each player is associated a win-loss objective. That is, the objective of each

player is specified in terms of a set $\Phi^i \subseteq \mathit{Plays}^i(\mathcal{G})$. We say a play $\rho \in \mathit{Plays}(\mathcal{G})$ is winning for player i if $\mathit{view}^i(\rho) \in \Phi^i$. We are interested in analyzing regular objectives of players and therefore assume that Φ^i for each player i can be presented in terms of a deterministic Muller automaton $\mathcal{M}^i$.

For a subset of players $\mathcal{C} \subseteq N$, a joint $\mathcal{C}$-objective $\Phi^{\mathcal{C}} \subseteq \mathit{Plays}^{\mathcal{C}}(\mathcal{G})$. When $\mathcal{C} = N$, the set of all players, the objective $\Phi^{\mathcal{C}}$ is simply a subset of $\mathit{Plays}(\mathcal{G})$.

Given a joint $\mathcal{C}$-objective for a subset of players $\mathcal{C}$, we say that a joint $\mathcal{C}$-strategy $\tau^{\mathcal{C}}$ ensures $\Phi^{\mathcal{C}}$ iff for all paths ρ consistent with $\tau^{\mathcal{C}}$, we have $\mathit{view}^{\mathcal{C}}(\rho) \in \Phi^{\mathcal{C}}$. A distributed $\mathcal{C}$-strategy $\mu^{\mathcal{C}}$ ensures $\Phi^{\mathcal{C}}$ if the joint $\mathcal{C}$-strategy $\tau^{\mathcal{C}}$ induced by $\mu^{\mathcal{C}}$ ensures $\Phi^{\mathcal{C}}$. The case when $\mathcal{C}$ constitutes a single player i, this amounts to saying that for all paths ρ consistent with μ^i, we have $\mathit{view}^i(\rho) \in \Phi^i$. Or in other words, μ^i is a winning strategy for objective Φ^i.

For two player games, we say the pair (Φ^1, Φ^2) defines a zero sum objective if the set $X = \{\rho \in \mathit{Plays}(\mathcal{G}) \mid \mathit{view}^1(\rho) \in \Phi^1\}$ and $Y = \{\rho \in \mathit{Plays}(\mathcal{G}) \mid \mathit{view}^2(\rho) \in \Phi^2\}$ satisfies the condition that $Y = \mathit{Plays}(\mathcal{G}) \setminus X$. The class of two player zero sum games defined in terms of local game arenas with public announcements need not be determined as illustrated by the following example.

Example 6.1.4 Consider the global game arena given in Figure 6.2. Let the objectives of players be as follows:

- Φ^1 is the set of all paths in $\mathit{Plays}^1(\mathcal{G})$ which cycles in the local state w_3^1 or w_6^1 of player 1.
- Φ^2 is the set of all paths in $\mathit{Plays}^2(\mathcal{G})$ which cycles in the local state w_4^2 or w_5^2 of player 2.

It can be easily seen that the pair (Φ^1, Φ^2) defines a zero sum objective. Player 1 does not have a winning strategy for Φ^1. To see this, suppose player 1 chooses w_1^1 at the game position (w_0^1, w_0^2) then there is a path where player 2 chooses w_2^2 which does not satisfy Φ^1. If player 1 chooses w_2^1 at (w_0^1, w_0^2) then the choice w_1^2 of player 2 leads to a path which does not satisfy Φ^2. Similarly player 2 does not have a strategy to ensure Φ^2 either, since she cannot distinguish between the global states (w_1^1, w_0^2) and (w_2^1, w_0^2). In other words, the game is not determined.

□

6.2 The verification question

Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, a subset of players $\mathcal{C} \subseteq N$ and a regular win-loss $\mathcal{C}$-objective $\Phi^{\mathcal{C}}$, the verification question asks:

- does there exists a distributed strategy $\mu^{\mathcal{C}}$ such that $\mu^{\mathcal{C}}$ ensures $\Phi^{\mathcal{C}}$?

In this section we show that the verification question is decidable in the proposed game model where players communicate by means of public announcements. When $\mathcal{C}$ consists of a single player i, the verification question asks whether there exists a strategy for player i to ensure outcome Φ^i. We first show that this question is decidable.

Techniques adopted for computing winning strategies in games of perfect information cannot be directly applied in this context. To see why, consider the verification question for a single player i in the perfect information setting. Let $\mathcal{M}^i$ be the deterministic Muller automaton representing the regular objective Φ^i. To solve the verification question we consider the tree unfolding $T_{\mathcal{G}}$ of the global game arena $\mathcal{G}$. We can build a tree automaton which at every node of player i guesses a choice and branches out on choices of nodes not belonging to i. $T_{\mathcal{G}}$ also runs $\mathcal{M}^i$ in parallel and verifies that all runs conform to the specification. An accepting run of this automaton would be the strategy for player i to ensure objective Φ^i.

In the case of incomplete information note that the guesses made by the tree automaton at various i nodes need to preserve the consistency requirement of the strategy ($\mathcal{S}3$). As noted by [KV99] this condition is non-regular and thus cannot be directly maintained by an automaton. [KV99] suggests a technique to circumvent this problem in terms of a construction using alternating tree automata. Here we show that the construction used in [KV99] can be appropriately modified to solve the verification question.

Alternating tree automata: For a finite set Υ of directions, a Υ-tree is a set $T \subseteq \Upsilon^*$ such that the following condition holds:

- if $s \cdot u \in T$ where $s \in \Upsilon^*$ and $u \in \Upsilon$ then $s \in T$ as well

Given a set Σ, a Σ labelled Υ tree is a pair $(T, \mathfrak{l})$ where T is a Υ tree and $\mathfrak{l} : T \to \Sigma$, i.e. each node of T is labelled with an element in Σ.

Alternating tree automata generalize nondeterministic tree automata and were first introduced in [MS87]. While a nondeterministic automaton can guess a set of

successor states and send one copy of itself along the subtrees rooted at its children, an alternating automaton can propagate several copies to a single child.

For a set X let $\mathcal{B}^+(X)$ denote the set of positive boolean formulas formed from the elements in X. That is, $\mathcal{B}^+(X) := \mathit{True} \mid \mathit{False} \mid x \in X \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2$. For a formula $\alpha \in \mathcal{B}^+(X)$ and a subset $Y \subseteq X$, we say Y satisfies α iff assigning *True* to elements in Y and *False* to elements in $X \setminus Y$ makes α true.

An alternating tree automaton over Σ-labelled Υ tree is $\mathcal{T} = (Q, \delta, q_0, F)$ where Q is a finite set of states, q_0 is the initial state, F defines the acceptance condition (a condition that defines a subset of Q^ω). The transition relation δ is a map $\delta : Q \times \Sigma \times 2^\Upsilon \to \mathcal{B}^+(Q \times \Upsilon)$ which satisfies the condition: if $q \in Q$, $a \in \Sigma$, $C \subseteq \Upsilon$ then $\delta(q, a, C)$ is a boolean formula in $\mathcal{B}^+(Q \times C)$.

Let $(T, \mathfrak{l})$ be a Σ-labelled tree, a run of the automaton $\mathcal{T}$ over $(T, \mathfrak{l})$ is a $(T \times Q)$ labelled tree (T_r, r) in which the root is labelled by q_0 and a label of a node u in T_r being (s, q) represents that the run at that node is reading the node s of the tree T and is in state q. Formally, (T_r, r) satisfies the following conditions:

- for the root node y_0, $r(y_0) = (\epsilon, q_0)$
- Let $y \in T_r$ and $r(y) = (s, q)$. Let C be the successor directions of the node s and $\delta(q, \mathfrak{l}(x), C) = \theta$ where θ is a formula in $\mathcal{B}^+(Q \times C)$. Let $Y \subseteq Q \times C$ be the set of all (q', c') such that there is a child y' of y with $r(y') = (q', x \cdot c')$. Then we require that Y satisfies the formula θ.

A path ρ of the run is said to be accepting if the sequence of Q-components of the labels of ρ satisfies the acceptance condition. The run (T_r, r) is accepting if all paths in it are accepting.

Solving the verification question: Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, let $\mathcal{G} = (W, \rightarrow, \mathbf{w}_0)$ be the induced global arena. For a player $i \in N$, we use the notation W^{-i} to denote the set $W^1 \times \ldots \times W^{i-1} \times W^{i+1} \times \ldots \times W^n$. For $\mathbf{w} \in W$, we use $\mathbf{w}^{-i}$ to denote the tuple $\mathbf{w}^{-i} = (w^1, \ldots, w^{i-1}, w^{i+1}, \ldots, w^n)$. For the global arena $\mathcal{G}$, the tree unfolding $T_\mathcal{G}$ (as given in Definition 6.1.2) can be viewed as a $W^i \times W^{-i} \times \widetilde{\Gamma}$ tree. A strategy tree t_{μ^i} for player i is then a W^i labelled $W^i \times W^{-i} \times \widetilde{\Gamma}$ tree $(T_\mathcal{G}, \mathfrak{l})$.

Definition 6.2.1 *For a global arena $\mathcal{G} = (W, \rightarrow, \mathbf{w}_0)$ we define $Proj(\mathcal{G}, W^{-i}) = (U, \rightarrow_P, u_0)$ where $U = W^i \times \widetilde{\Gamma}$ and $u_0 = view^i(\mathbf{w}_0)$. The move relation $\rightarrow_P \subseteq U \times U$ is defined as:*

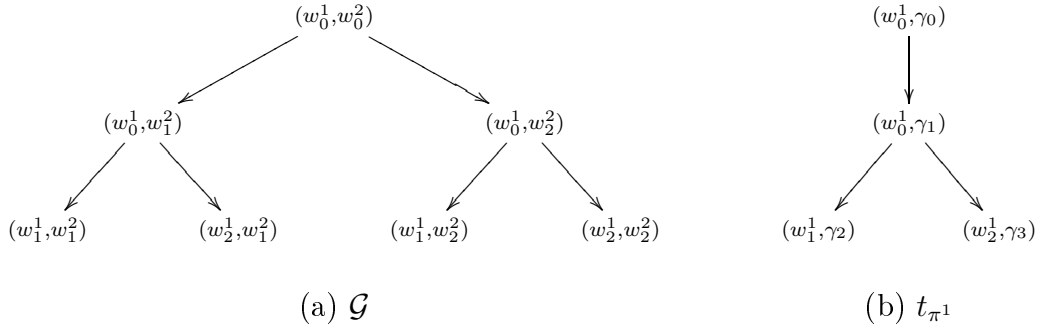
- $u \rightarrow_P u'$ *iff there exists* $\mathbf{w}, \mathbf{v} \in W$ *such that* $view^i(\mathbf{w}) = u$, $view^i(\mathbf{v}) = u'$ *and* $\mathbf{w} \rightarrow \mathbf{v}$

It can be easily verified that for all paths ρ in $\mathcal{G}$, $view^i(\rho)$ is a path in $Proj(\mathcal{G}, W^{-i})$. $Proj(\mathcal{G}, W^{-i})$ is an arena obtained by projecting out from $\mathcal{G}$ the components which give rise to imperfect information for player i in $\mathcal{G}$. A strategy π^i of player i in $Proj(\mathcal{G}, W^{-i})$ is therefore a function $\pi^i : (W^i \times \widetilde{\Gamma})^* \rightarrow W^i$ such that $\pi^i(\epsilon) = w_0^i$. A strategy π^i in $Proj(\mathcal{G}, W^{-i})$ generates a strategy μ^i in $\mathcal{G}$ as follows: for all partial plays $\varrho \in \mathfrak{P}(\mathcal{G})$, $\mu^i(\varrho) = \pi^i(view(\varrho))$. By definition, the generated strategy μ^i satisfies the condition $(\mathcal{S}3)$.

Let $\mathcal{G}' = Proj(\mathcal{G}, W^{-i})$. Given a W^i labelled $W^i \times \widetilde{\Gamma}$ strategy tree $t_{\pi^i} = (T_{\mathcal{G}'}, \mathfrak{l}')$, we define $wide_{\mathcal{G},W^{-i}}(t_{\pi^i})$ as the W^i labelled $W^i \times W^{-i} \times \widetilde{\Gamma}$ tree $t = (T_{\mathcal{G}}, \mathfrak{l})$ which satisfies the condition:

- for every node s in $T_{\mathcal{G}}$ we have $\mathfrak{l}(s) = \mathfrak{l}'(view^i(s))$.

Thus a strategy tree t_{π^i} in $Proj(\mathcal{G}, W^{-i})$ generates a strategy tree t_{μ^i} in $\mathcal{G}$ by the transformation $t_{\mu^i} = wide_{\mathcal{G},W^{-i}}(t_{\pi^i})$.



(a) $\mathcal{G}$ (b) t_{π^1}

Figure 6.3: Game arena and strategy tree

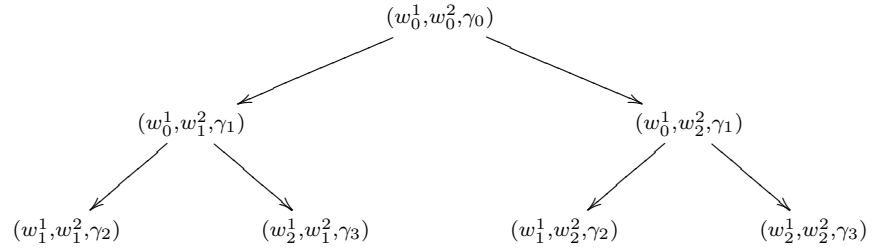


Figure 6.4: Strategy tree in $\mathcal{G}$

Example 6.2.2 To illustrate the widening operator, consider the game arena $\mathcal{G}$ given in Figure 6.3(a). Let t_{π^1} be the structure shown in Figure 6.3(b) where $\mathfrak{l}'(w_0^1, \gamma_0) = w_0^1$, $\mathfrak{l}'(w_0^1, \gamma_1) = w_1^1$. The structure $wide_{\mathcal{G},W^{-i}}(t_{\pi^i})$ is shown in Figure 6.4. Since $view^1(w_0^1, w_1^2, \gamma_1) = view^1(w_0^1, w_2^2, \gamma_1)$ we have $\mathfrak{l}(w_0^1, w_1^2, \gamma_1) = \mathfrak{l}(w_0^1, w_2^2, \gamma_1) = w_1^1$. □

Definition 6.2.3 *For a global arena $\mathcal{G}$, a* choice function *of player i, is a map $\mathfrak{c}^i : W^* \to W^i$ which satisfies the following conditions:*

- $\mathfrak{c}^i(\epsilon) = w_0^i$.
- *for all finite partial plays $\varrho \in \mathfrak{P}(\mathcal{G})$, if $\mathfrak{c}^i(\varrho) = w^i$ then there exists $\mathbf{v}$ such that $last(\varrho) \rightarrow \mathbf{v}$ and $v^i = w^i$.*

In other words, a choice function for player i is similar to a strategy except that the condition $(\mathcal{S}3)$ need not hold. A choice function which is consistent with the information partition for player i constitutes a strategy for the player. A choice tree for player i is defined in a manner similar to that of strategy tree, i.e. it is a W^i labelled $W^i \times W^{-i} \times \widetilde{\Gamma}$ tree.

Lemma 6.2.4 *Given a global game arena $\mathcal{G}$, a player $i \in N$ and objective Φ^i, we can construct an alternating tree automaton $\mathcal{T}_{\Phi^i}$ accepting the set of all choice trees for player i in $\mathcal{G}$ which ensure Φ^i.*

Proof: Let $\mathcal{M}^i$ denote the Muller automaton corresponding to Φ^i. The automaton $\mathcal{T}_{\Phi^i}$ ensures that the structure t constitutes the unfolding of the game arena $\mathcal{G}$. The labelling function associated with t specifies the choice function of player i. The automaton also runs $\mathcal{M}^i$ in parallel and ensures that all the paths which is consistent with the labelling is accepted by $\mathcal{M}^i$. □

Lemma 6.2.5 *Given an alternating tree automaton $\mathcal{T}$ over W^i labelled $W^i \times W^{-i} \times \Gamma$ trees, we can construct an alternating tree automaton $\mathcal{T}'$ over W^i labelled $W^i \times \Gamma$ trees such that $\mathcal{T}'$ accepts a labelled tree t' iff $\mathcal{T}$ accepts $wide_{(\mathcal{G},W^{-i})}(t')$.*

Proof: Let $\mathcal{T} = (Q, \delta, q_0, F)$. We define $\mathcal{T}' = (Q \times W^{-i}, \delta', \langle q_0, w_0^{-i} \rangle, F \times W^{-i})$. The transition $\delta'(\langle q, w^{-i} \rangle, u^i)$ is obtained from $\delta(q, u^i)$ by replacing each element $((u^i, u^{-i}, \gamma'), q')$ by the element $((u^i, \gamma'), \langle q', u^{-i} \rangle)$. We show that for all W^i labelled $W^i \times \widetilde{\Gamma}$ trees $t' = (T'_{\mathcal{G}}, \mathfrak{l}')$ we have $t' \in Lang(\mathcal{T}')$ iff $wide_{(\mathcal{G},W^{-i})}(t') \in Lang(\mathcal{T})$.

($\Leftarrow$): Suppose $wide_{(\mathcal{G},W^{-i})}(t') \in Lang(\mathcal{T})$, let (T_r, r) be an accepting run of $\mathcal{T}$. We have $r : T_r \rightarrow ((W^i \times W^{-i} \times \widetilde{\Gamma})^* \times Q)$. Consider the tree (T_r, r') where for all $u \in T_r$ with $r(u) = (s, q)$ and $\mathbf{w} = last(s)$ we have $r'(u) = (view^i(s), (q, w^{-i}))$. (T_r, r') is an accepting run of $\mathcal{T}'$ on t'.
($\Rightarrow$): Suppose $t' \in Lang(\mathcal{T}')$, let (T_r, r') be an accepting run of $\mathcal{T}'$. We have $r' : T_r \rightarrow (W^i \times \widetilde{\Gamma})^* \times (Q \times W^{-i})$. Consider the tree (T_r, r) where r is a map $r : T_r \rightarrow (W^i \times W^{-i} \times \widetilde{\Gamma})^* \times Q$ is defined as:

- $r(\epsilon) = r'(\epsilon)$
- For $u \cdot d \in T_r$ with $r(u) = (s, q')$ and $r'(u \cdot d) = (s' \cdot (w^i, \gamma), \langle q, w^{-i} \rangle)$, we have $r(u \cdot d) = (s \cdot \langle w^i, w^{-i}, \gamma \rangle, q)$.

(T_r, r) is an accepting run of $\mathcal{T}$ on $wide_{(\mathcal{G},W^{-i})}(t')$. □

Let the automaton constructed in Lemma 6.2.5 be denoted as $narrow_{(\mathcal{G},W^{-i})}(\mathcal{T})$.

Lemma 6.2.6 *Given a global arena $\mathcal{G}$, a player $i \in N$ and objective Φ^i, there exists a strategy for player i to ensure Φ^i iff $narrow_{(\mathcal{G},W^{-i})}(\mathcal{T}_{\Phi^i}) \neq \emptyset$.*

Proof:
($\Rightarrow$): Suppose there exists a strategy π^i which ensures Φ^i, from Lemma 6.2.4 we have $wide_{(\mathcal{G},W^{-i})}(t_{\pi^i}) \in Lang(\mathcal{T}_{\Phi^i})$. From Lemma 6.2.5 we get $t_{\pi^i} \in Lang(narrow_{(\mathcal{G},W^{-i})}(\mathcal{T}_{\Phi^i}))$.
($\Leftarrow$): Suppose $Lang(narrow_{(\mathcal{G},W^{-i})}(\mathcal{T}_{\Phi^i}))$ is not empty. Then there exists a tree $t_{\pi^i} \in Lang(narrow_{(\mathcal{G},W^{-i})}(\mathcal{T}_{\Phi^i}))$. By Lemma 6.2.5 we have $wide_{(\mathcal{G},W^{-i})}(t_{\pi^i}) \in Lang(\mathcal{T}_{\Phi^i})$. From Lemma 6.2.4 we get there exists a strategy for player i to ensure Φ^i. □

Proposition 6.2.7 then follows from Lemma 6.2.6.

Proposition 6.2.7 *Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, a player $i \in N$ and an objective Φ^i, it is decidable to check if there exists a strategy μ^i for player i such that μ^i ensures Φ^i.*

In the above construction, the use of alternating tree automata is mainly for convenience. One could presumably work with nondeterministic automata as well. However, such a construction would typically involve the size of the resulting automaton to be exponential. In contrast, alternation provides a helpful mechanism whereby the description of the automaton itself is very simple and all the combinatorial difficulty is shifted to the non-emptiness test.

Theorem 6.2.8 *Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, a subset of players $\mathcal{C}$ and an objective $\Phi^{\mathcal{C}}$, it is decidable to check whether there exists a distributed strategy $\mu^{\mathcal{C}}$ such that $\mu^{\mathcal{C}}$ ensures $\Phi^{\mathcal{C}}$.*

Proof: From Proposition 6.2.7 it follows that it is decidable to check whether $\mathcal{C}$ has a joint strategy $\tau^{\mathcal{C}}$ which ensures $\Phi^{\mathcal{C}}$. We show that the existence of a joint strategy implies the existence of a distributed strategy. Since existence of a distributed strategy trivially implies the existence of a joint strategy, the theorem then follows.
Claim : For all joint $\mathcal{C}$ strategies $\tau^{\mathcal{C}}$, for all $\varrho, \varrho' \in \mathfrak{P}(\mathcal{G})$ consistent with $\tau^{\mathcal{C}}$ where $\mathcal{C} \cap \mathit{enabled}(\varrho) \neq \emptyset$, $\mathit{view}^{\mathcal{C}}(\varrho) \neq \mathit{view}^{\mathcal{C}}(\varrho')$ implies $\forall i \in \mathcal{C}$ such that $i \in \mathit{enabled}(\varrho)$ we have $\mathit{view}^i(\varrho) \neq \mathit{view}^i(\varrho')$.

The claim can be verified as follows. Suppose it is not true, then there exists a player $i \in \mathcal{C}$ and sequences ϱ and ϱ' such that $\mathit{view}^i(\varrho) = \mathit{view}^i(\varrho')$. If $\varrho = \varrho'$ then the claim follows easily. It cannot be the case that ϱ is a strict prefix of ϱ' or vice-versa. Let ϱ_1 be the maximum common prefix of ϱ and ϱ'. Let $\varrho = \varrho_1 \cdot \mathbf{w}_2 \cdot \mathbf{w}_3 \cdots \mathbf{w}_m$ and $\varrho' = \varrho_1 \cdot \mathbf{v}_2 \cdot \mathbf{v}_3 \cdots \mathbf{v}_m$. Since $\mathit{view}^i(\varrho) = \mathit{view}^i(\varrho')$, it follows that $\mathit{view}^i(\varrho_1 \cdot \mathbf{w}_2) = \mathit{view}^i(\varrho_1 \cdot \mathbf{v}_2)$. Since announcements are public, $\chi(\mathbf{w}_2) = \chi(\mathbf{v}_2) = \gamma$. Since ϱ and ϱ' are consistent with $\tau^{\mathcal{C}}$, we have $w_2^{\mathcal{C}} = v_2^{\mathcal{C}}$. Since announcements are public we have $\mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{w}_2) = \mathit{view}^{\mathcal{C}}(\varrho_1) \cdot (w_2^{\mathcal{C}}, \gamma) = \mathit{view}^{\mathcal{C}}(\varrho_1) \cdot (v_1^{\mathcal{C}}, \gamma) = \mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{v}_2)$. We have the following two cases.

- $\mathcal{C} \cap \mathit{enabled}(\varrho_1 \cdot \mathbf{w}_2) = \emptyset$: Since announcements are public, we get $\mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{w}_2 \cdot \mathbf{w}_3) = \mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{v}_2 \cdot \mathbf{v}_3)$.
- $\mathcal{C} \cap \mathit{enabled}(\varrho_1 \cdot \mathbf{w}_2) \neq \emptyset$: This also implies that $\mathcal{C} \cap \mathit{enabled}(\varrho_1 \cdot \mathbf{v}_2) \neq \emptyset$ and $\tau^{\mathcal{C}}(\varrho_1 \cdot \mathbf{w}_2) = \tau^{\mathcal{C}}(\varrho_1 \cdot \mathbf{v}_2)$. Again since announcements are public we get that $\mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{w}_2 \cdot \mathbf{w}_3) = \mathit{view}^{\mathcal{C}}(\varrho_1 \cdot \mathbf{v}_2 \cdot \mathbf{v}_3)$.

Proceeding in this manner we get $\mathit{view}^{\mathcal{C}}(\varrho) = \mathit{view}^{\mathcal{C}}(\varrho')$ which is a contradiction.

End of claim

From the above claim it follows that for any partial play $\varrho \in \mathfrak{P}(\mathcal{G})$, for any player i whose move is enabled at ϱ, given $\mathit{view}^i(\varrho)$, player i can make exactly the same move that is dictated by the joint strategy. In this way we define all the local strategies. □

6.3 Games with private communication

In the game model introduced above all communication between players are due to public announcements. Communication through private channels between players can be captured by modifying the structure of the announcement alphabet.

For each player i we have a set of communication alphabets $\{\Gamma^i_j\}_{j\in N}$ with the interpretation that Γ^i_j is the set of symbols which player i can announce to player j. Let $\Gamma = \bigcup_{i,j\in N}\Gamma^i_j$. Let $\Theta^i_{put} = \{\nu^i : N \to \Gamma \mid \forall j \in N, \nu^i(j) \in \Gamma^i_j\}$ and $\Theta^i_{get} = \{\eta^i : N \to \Gamma \mid \forall j \in N, \eta^i(j) \in \Gamma^j_i\}$. The function ν^i specifies the announcements made by player i and η^i specifies the announcements received by player i.

The local game arena for player i is then given by $\mathcal{G}^i = (W^i, \to_i, w^i_0)$ where W^i is the set of local game positions, w^i_0 is the initial game position and $\chi^i : W^i \to \Theta^i_{put}$. The move relation $\to_i : W^i \times \Theta^i_{get} \to 2^{W^i}$ satisfies the following condition: for all $w^i, v^i \in W^i$

- $w^i \xrightarrow{\eta^i} v^i$ implies $\eta^i(i) = \chi(w^i)(i)$.

The global game arena $\mathcal{G} = (W, \to, \mathbf{w}_0)$ is derived from the local arenas as in the earlier case. The global game positions are $W = W^1 \times \cdots \times W^n$ and $\mathbf{w}_0 = (w^1_0, \ldots, w^n_0)$. The move relation $\to \subseteq W \times W$ which satisfies the following property: $\mathbf{w}\to\mathbf{v}$ iff for all $i, j \in N$

- for all $i \in \mathit{enabled}(\mathbf{w})$, $w^i \xrightarrow{\eta^i}_i v^i$ where $\eta^i(j) = \chi^j(w^j)(i)$ or $w^i = v^i$.
- for all $i \notin \mathit{enabled}(\mathbf{w})$, $w^i = v^i$.

6.3.1 Undecidability of the verification question

In this section we show that if we look at the class of imperfect information games where private communication between players are allowed then the verification question becomes undecidable. The interesting fact is that even for simple reachability objectives, the problem remains undecidable. We prove undecidability by giving a reduction to the Post's correspondence problem. The proof proceeds along the lines of the one presented in [Ber06] which shows that distributed strategy synthesis is undecidable for the distributed games model introduced in [MW03] with reachability objectives.

Post's correspondence problem: An instance of Post's correspondence problem (PCP) consists of two lists, $A = x_1, \ldots, x_k$ and $B = y_1, \ldots, y_k$ of strings over some finite alphabet set Z. This instance of PCP has a solution if there is any sequence of integers $j_1, j_2, \ldots, j_m$ with $m \geq 1$ such that $x_{j_1}, x_{j_2}, \ldots, x_{j_m} = y_{j_1}, y_{j_2}, \ldots, y_{j_m}$. The sequence $j_1, j_2, \ldots, j_m$ is said to be a solution to this instance of PCP.

	List A	List B
j	x_j	y_j
1	1	111
2	10111	10
3	10	0

Figure 6.5: A PCP instance

Example 6.3.1 Let $Z = \{0, 1\}$, let A and B be the list of strings as given in Figure 6.5. Consider the indices $j_1 = 2$, $j_2 = 1$, $j_3 = 1$ and $j_4 = 3$. Then we have $x_2x_1x_1x_3 = y_2y_1y_1y_3 = 101111110$. Thus the sequence of indices mentioned above is a solution. □

Theorem 6.3.2 ([HU79]) *PCP is undecidable.*

Reachability objective: Given a game $G = \{\mathcal{G}^i\}_{i \in N}$, to model reachability objectives, we assume that for each player $i \in N$, there is a state $reach^i$ such that for all $j \in N$, $\chi^i(reach^i)(j) = reach^i$. That is, once a player enters $reach^i$ then she announces this fact to all the other players. The *reach* state is also a sink state for players and once entered, the player remains in this state. The reachability objective Φ^i of player i can then be given as $\Phi^i = \{(w_0^i, \gamma_0)(w_1^i, \gamma_1) \ldots \in Plays^i(\mathcal{G}) \mid \exists j \text{ with } w_j^i = reach^i\}$.

Theorem 6.3.3 *The verification question is undecidable for reachability games with private communication.*

Proof: Given an instance of PCP, we construct a game such that this instance of PCP has a solution iff the verification question is decidable for the constructed game. Let the instance of the PCP be given by the list of strings $A = x_1, \ldots, x_k$ and $B = y_1, \ldots, y_k$ over the set of alphabets Z. We assume that the state space of the players is rich enough to code up the PCP instance. That is, for each index

$j \in \{1, \ldots, k\}$ there is a state which represents the index, for each alphabet in Z, there is a state which identifies the alphabet and the lists A and B are also coded into the state space. We construct a game consisting of four players where each of the players' functions are as follows:

- Player 0 has perfect information about the game. That is, Players 1, 2 and 3 convey their exact local states to Player 0 through the private communication channel. Player 0 makes the choice of the list (either A or B) and also schedules the moves of other players.
- Player 1, whenever her move is enabled, chooses indices $j \in \{1, \ldots, k\}$. At any point she can also choose to enter the $quit^1$ state indicating the end of the choice of indices.
- Player 2 chooses letters from the alphabet Z. At any point Player 2 can also choose to enter the $quit^2$ state indicating the end of choice of strings generated.
- Player 3 is a deterministic program whose function is to match the index j chosen by Player 1 with the letters generated by Player 2 and make sure that it is in fact the j^{th} string in the list chosen by Player 0.

The game proceeds as follows: initially Player 0 moves, chooses one of the list (either A or B) and communicates the choice to Player 3 through the private communication channel. The choice of list is not revealed to Players 1 and 2. Next Player 1 is scheduled to move, she chooses an index $j \in \{1, \ldots, k\}$ and communicates the choice to both Player 0 and 3. Player 2 is unaware of the choice made by Player 1. Player 2 now chooses letters from Z and communicates the choice to Players 0 and 3. Player 1 is unaware of the choice of Player 2. Player 3 matches the string generated by Player 2 and ensures that it matches the j^{th} string in the list chosen by Player 0. If it does not match then Player 3 enters a *reject* state. Note that Player 3 need not keep track of the entire string in order to do this. The lists A and B are encoded in the state space of Player 3 and she just needs to match the sequence of alphabets chosen by Player 2 with the j^{th} string in the appropriate list. This can be achieved with finite memory. Once Player 3 enters the *reject* state then the game stays in a sink state and does not proceed any further. If the string matches then Player 1 chooses another index and the game goes on as described above.

Players 1 and 2 can choose to enter their *quit* states at any time when they are scheduled. Since *quit* states are intended to capture the end of the sequence of indices or strings, Players 1 and 2 are required to enter their quit states in their successive moves. If at any round Player 1 enters its *quit* state and Player 2 continues to choose letters from Z then Player 3 enters its reject state. Similarly if Player 2 enters its quit state while Player 1 has not yet chosen to quit then Player 3 enters its reject state. Player 3 also ensures that Players 1 and 2 do not choose *quit* states in their initial moves. If both Player 1 and Player 2 enter their respective quit states in succession, then Player 3 enters an *accept* state which is then communicated to all the players. If Player 3 ever enters its *accept* state then both Players 1 and 2 move to states $reach^1$ and $reach^2$ respectively.

Now consider the subset of players $\mathcal{C} = \{1, 2\}$ and the joint $\mathcal{C}$ objective $\Phi^{\mathcal{C}}$ be the reachability objective where both players enter the states $reach^1$ and $reach^2$ respectively. We claim that for a distributed strategy (μ^1, μ^2) to exist which ensures $\Phi^{\mathcal{C}}$, the PCP instance needs to have a solution. To see this, suppose the PCP instance does not have a solution, then Player 0 can appropriately choose the lists A or B so that the resulting run does not ensure objective $\Phi^{\mathcal{C}}$. For instance, suppose Player 1 generates the sequence of indices $j_1, \ldots, j_m$ and then enters the state $quit^1$ and Player 2 generates the string $x_{j_1} x_{j_2} \ldots x_{j_m}$ and enters the state $quit^2$. Now consider the run where Player 0 chooses the list B. Since the PCP instance does not have a solution, $x_{j_1} x_{j_2} \ldots x_{j_m} \neq y_{j_1} y_{j_2} \ldots y_{j_m}$. Let p be the first index where the strings differ and let p occur in the substring y_{i_l}. Now consider what happens in round i_l: Player 1 chooses the index i_l, however, since $x_p \neq y_p$, the string of alphabets generated by Player 2 in round i_l does not match with y_{i_l}. Player 3 therefore enters the reject state and thus the objective $\Phi^{\mathcal{C}}$ is not satisfied.

If the instance of PCP has a solution, then let the indices $j_1, j_2, \ldots, j_m$ be the solution. Consider the strategies where Player 1 chooses indices $j_1, j_2, \ldots, j_m$ and enters $quit^1$, Player 2 generates the string $x_{j_1} x_{j_2} \ldots x_{j_m}$ and enters $quit^2$. Since $x_{j_1} x_{j_2} \ldots x_{j_m} = y_{j_1} y_{j_2} \ldots y_{j_m}$, irrespective of whether Player 0 chooses the list A or B, Player 3 eventually enters the *accept* state. It then follows that the subset of players $\mathcal{C} = \{1, 2\}$ has a distributed strategy to ensure objective $\Phi^{\mathcal{C}}$ for the above mentioned games iff the instance of PCP has a solution. □

6.4 Games with overlapping objectives

In the previous sections we looked at games where players have win-loss objectives. In general, players' objectives are specified in terms of a preference ordering $\preceq^i \subseteq \mathit{Plays}^i(\mathcal{G}) \times \mathit{Plays}^i(\mathcal{G})$. Once the preference orderings of players are defined, we can look at notions like best response and equilibrium as defined in Section 2.1.3. For the purpose of algorithmic analysis, the preference ordering of each player i can be presented in a finite fashion in terms of an evaluation automaton $\mathcal{E}^i = (\mathcal{M}^i, \lhd^i)$ where $\mathcal{M}^i = (R^i, \Delta^i, r_0^i, \mathcal{F}^i)$ is the underlying Muller automaton (see Definition 5.1.2). Since we want the evaluation automaton to induce a preference ordering over paths in $\mathit{Plays}^i(\mathcal{G})$ we take the transition function $\Delta^i : R^i \times W^i \times \widetilde{\Gamma} \to R^i$.

Bounded memory strategies of players can also be presented in terms of deterministic advice automata (Definition 2.3.3) by appropriately modifying the transition function to take into account the views of players.

In the context of non-zero sum games, we show that the best response computation can be effectively performed.

Theorem 6.4.1 *Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, preference orderings $\{\preceq^i\}_{i \in N}$ of players and a strategy profile μ^{-i} in terms of advice automata, the best response for player i can be effectively computed.*

Proof: Let the strategy profile μ^{-i} be presented as advice automata $\mathcal{A}^{-i} = (\mathcal{A}^1, \ldots, \mathcal{A}^{i-1}, \mathcal{A}^{i+1}, \ldots, \mathcal{A}^n)$. Let the evaluation automaton for player i be $\mathcal{E}^i = (\mathcal{M}^i, \lhd^i)$ where $\mathcal{M}^i = (R^i, \Delta^i, r_0^i, \mathcal{F}^i)$. For each $F \in \mathcal{F}^i$, we can construct a nondeterministic automaton A_F which explores paths of $\mathcal{G}$ as follows. It consults $\mathcal{A}^{-i}$ to pick moves of players $j \in N \setminus \{i\}$ and simply guesses i's moves. Since strategies of other players are *deterministic* and the initial game position is unique, this defines a unique path in the arena. Automaton A_F runs the win-loss evaluation automaton $\mathcal{E}_F^i$ for player i in parallel and checks if the run is winning for player i. Now, we can enumerate the $F \in \mathcal{F}$ in such a way that those higher in $\lhd^i$ appear earlier in the enumeration. We try automata A_F in this order. □

We then have the following corollary.

Corollary 6.4.2 *Given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$, preference orderings $\{\preceq^i\}_{i \in N}$ of players and a strategy profile $\overline{\mu}$ in terms of advice automata it is possible to check whether the strategy profile constitutes a Nash equilibrium.*

6.5 Discussion

In the context of non-zero sum games of imperfect information, we showed that the best response computation can be done and it is possible to verify whether a given strategy profile constitutes a Nash equilibrium. It turns out that Nash equilibrium need not always exist in such games. In fact, a game similar to the one given in Example 6.1.4 can be used to show this fact. A natural question therefore would be to ask: given a game structure $\mathbb{G} = \{\mathcal{G}^i\}_{i \in N}$ along with preference orderings $\{\preceq^i\}_{i \in N}$ of players,

- is it decidable to check if Nash equilibrium exists in $\mathbb{G}$?

An obvious approach would be to try using techniques similar to the LAR tree construction developed in Chapter 5 to tackle this question. Unfortunately the finite tree unfolding construction does not solve the problem, since in general, the strategies constructed need not respect the information partition of players. However, the core idea of the LAR construction is to transform the game structure $\mathcal{G}$ into a bigger structure $\mathcal{G}'$ which satisfies the property:

- there exists an equilibrium profile in $\mathcal{G}$ iff there exists an equilibrium profile in memoryless strategies in $\mathcal{G}'$.

We believe that using this approach one can show that the above mentioned question is decidable for imperfect information games where players communicate through public announcements. However, we do not have a proof of this fact yet.

Chapter 7

Conclusion

In conclusion, the main topic of this thesis has been the analysis of strategies in games. We have focussed our attention on two main aspects: the algorithmic analysis and the logical analysis of strategies. Algorithmic analysis of strategies included issues like,

- synthesis of winning strategies for two player zero sum games, and
- best response computation and synthesis of equilibrium profiles for non-zero sum games.

For finite extensive form games, our main tool for algorithmic analysis was the backward induction algorithm. For infinite duration games, on the other hand, even presenting the strategies and objectives of players in a finite fashion is a non-trivial issue. We proposed evaluation automata as a convenient finite state model to present the preference orderings of players in an infinite duration game. We showed that in the case when preference orderings of players are presented in terms of evaluation automata, the backward induction procedure can be effectively adapted to synthesize an equilibrium strategy profile.

Algorithmic analysis as mentioned above, analyses functional strategies of players. However, functional strategy synthesis even though theoretically possible, may not necessarily be a practical tool in terms of a prescriptive theory for players. In this context, we suggested that it makes sense to look at strategies as partially defined objects and proposed a logical syntax to represent strategies in terms of their observable properties. Thus strategy specifications formed our basis of logical analysis of strategies. We showed how strategy specifications can be embedded into a simple modal logic to reason about games. On the technical front, we showed that

the logic admits a complete axiomatization and that the model checking problem for the logic is decidable. We also considered how the logical analysis can be adapted in the situation where the game itself has compositional structure. We proposed a logic which explicitly takes into account the compositional structure of games for strategic reasoning. We showed that the logic admits a complete axiomatization and that its satisfiability problem is decidable.

As mentioned in the introduction, we do not consider the very important concept of mixed strategies in the logical analysis. Thus one of the natural extensions to strategy specifications is to incorporate the notion of expectations of players. To come up with prescriptive mechanisms which provide advice to players on how to play, it is essential to be able to represent a player's expectations about the behaviour of the opponent. The expectations need not necessarily be represented in a probabilistic manner and could also be based on abstract notions like "likelihood" [HR87]. Introducing expectations of players is particularly interesting in the framework of unbounded game composition as it allows players to learn from the past information, revise their expectations and accordingly make use of it to generate sophisticated plans. Enriching the framework to be able to represent expectations of players is thus a challenging exercise. A related work in this context is that of [AB95] which looks at the epistemic conditions of players in terms of equilibrium notions.

All the above comments were regarding games of perfect information. For games of imperfect information, algorithmic as well as logical analysis remains in its early stages of development and most of the work which exists in the literature provide negative results. In the context of algorithmic analysis, even in finite extensive form games, the techniques developed for perfect information do not easily extend to games of imperfect information. For instance, our core technique of backward induction crucially relies on the fact that players have perfect information in making assumptions on how a rational opponent would play. For finite extensive form games, however, since the set of deterministic strategies of players is finite, one could enumerate all the strategies and decide on the existence of winning strategies as well as synthesis of equilibrium profiles.

The situation is less clear in the case of imperfect information games since in general stable strategy profiles in these games require memory; providing bounds on memory is a challenging task. In the case of two player zero sum games, it is possible to transform a game of imperfect information into a game of perfect information via

the subset construction such that the existence of winning strategies for at least one player is preserved. This construction was originally suggested by Reif in [Rei84]. Thus checking for existence of winning strategies in such games is decidable. For multi-player non-zero sum games, in general, even the question of checking whether an equilibrium profile exists turns out to be undecidable. A proof of this runs along the lines of the undecidability proof presented in Section 6.3.

As far as the logical analysis is concerned, for finite extensive form games, the modal and dynamic logic framework can be extended to reason about games with imperfect information. [Ben01] looks at reasoning about such games with respect to an epistemic modal language.

For unbounded duration games, there are hardly any decidable logics to reason about games with imperfect information. The ability of ATL to reason about games with imperfect information has been studied in the literature. The results are mostly negative; in [AHK02], the authors show that the ATL model checking problem for multiple players with imperfect information is undecidable. [Sch04] and [JvdH04] look at extensions of ATL which combine incomplete information and imperfect recall. [ÅW09] proposes an extension of ATL where bounded memory and bounded recall are explicitly taken into account in the logical language.

The notion of partial strategies makes sense in the context of a prescriptive theory for imperfect information games as well. Strategy specifications as introduced in Chapter 3 can be utilised to specify partial strategies in imperfect information games. As noted in [Ben07], imperfect information games encompass two intuitively different senses of uncertainty.

- "Future uncertainty": the uncertainty of players which arises from their lack of knowledge of what other players are going to do in the future.
- "Observation uncertainty": the uncertainty arising due to players not being able to observe events in the past.

Strategy specifications already incorporate the notion of future uncertainty of players. However, coming up with an appropriate logical language which embeds these specifications and can effectively reason about games with imperfect information is a challenging task. For it to be realistic, such a language also needs to incorporate epistemic attitudes and beliefs of players [HFMV95].

Bibliography

[AB95] R. J. Aumann and A. Brandenburger. Epistemic conditions for Nash equilibrium. *Econometrica*, 63:1161–1180, 1995.

[AD05] R. J. Aumann and J. H. Dreze. When all is said and done, how should you play and what should you expect? `http://www.ma.huji.ac.il/raumann/pdf/dp_387.pdf`, March 2005.

[Ågo06] T. Ågotnes. Action and knowledge in alternating time temporal logic. *Synthese*, 149(2):377–409, 2006.

[AHK02] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

[AR88] D. Abreu and A. Rubinstein. The structure of Nash equilibrium in repeated games with finite automata. *Econometrica*, 56:1259–1282, 1988.

[ÅW09] T. Ågotnes and D. Walther. A logic of strategic ability under bounded memory. *Journal of Logic Language and Information*, 18:55–77, 2009.

[Axe84] Robert Axelrod. *The Evolution of Cooperation*. New York: Basic Books, 1984.

[Ben01] J. van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[Ben02] J. van Benthem. Extensive games as process models. *Journal of Logic Language and Information*, 11:289–313, 2002.

[Ben06] J. van Benthem. Rationalization and promises in games. *Philosophical Trends, Supplement 2006*, pages 1–6, 2006.

[Ben07] J. van Benthem. In praise of strategies. In J. van Eijck and R. Verbrugge, editors, *Foundations of Social Software*, Studies in Logic, pages 283–317. College Publications, 2007.

[Ber03] D. Berwanger. Game logic is strong enough for parity games. *Studia logica*, 75(2):205–219, 2003.

[Ber05] D. Berwanger. *Games and Logical Expressiveness.* Ph.D. Thesis, Department of Computer Science, RWTH Aachen, Germany, 2005.

[Ber06] J. Bernet. *Discrete games for the synthesis and validation of communication processes.* PhD thesis, University of Bordeaux, 2006.

[Ber07] D. Berwanger. Admissibility in infinite games. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, LNCS. Springer-Verlag, 2007.

[BGL07] J. van Benthem, S. Ghosh, and F. Liu. Modelling simultaneous games with concurrent dynamic logic. In *A Meeting of Minds, Proceedings of the Workshop on Logic, Rationality and Interaction*, pages 243–258, 2007.

[BL69] J.R. Büchi and L.H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the Americal Mathematical Society*, 138:295–311, 1969.

[Bon01] G. Bonanno. Branching time logic, perfect information games and backward induction. *Games and Economic Behavior*, 36(1):57–73, 2001.

[Bon02] G. Bonanno. Modal logic and game theory: Two alternative approaches. *Risk Decision and Policy*, 7:309–324, December 2002.

[BOR06] J. van Benthem, S. van Otterloo, and O. Roy. Preference logic, conditionals, and solution concepts in games. *Uppsala Philosophical Studies*, 53, 2006.

[Bor07] S. Borgo. Coalitions in action logic. In *Proceedings IJCAI'07*, pages 1822–1827, 2007.

[Büc62] J.R. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the 1960 International Congress for Logic, Mehodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.

[Büc83] J.R. Büchi. State-strategies for games in $f_{\delta\sigma} \cap g_{\delta\sigma}$. *Journal of Symbolic Logic*, 48:1171–1198, 1983.

[Chu63] A. Church. Logic, arithmetics and automata. In *Proceedings of the International Congress of Mathematicians, 1962*, pages 23–25. Institut Mittag-Leffler, 1963.

[CJM04] K. Chatterjee, M. Jurdzinski, and R. Majumdar. On Nash equilibria in stochastic games. In *Proceedings of the 13th Conference of the European Association for Computer Science Logic*, volume 3210 of *LNCS*, pages 26–40. Springer, 2004.

[Ehr61] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49:129–141, 1961.

[Far02] B. Farwer. ω-automata. In *Automata, Logics and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.

[FL77] M.J. Fisher and R.E. Ladner. Propositional modal logic of programs. In *Proceedings of the 9th Symposium on Theory of Computing*, pages 286–294. ACM, 1977.

[FL79] M.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[FW94] L. Fortnow and D. Whang. Optimality and domination in repeated games with bounded players. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 741–749. ACM, 1994.

[Gal79] D. Gale. The game of hex and Brouwer fixed-point theorem. *The American Mathematical Monthly*, 86:818–827, 1979.

[GH82] Y. Gurevich and L. Harrington. Trees, automata and games. In *Proceedings of the 14th Symposium on Theory of Computing*, pages 60–65. ACM Press, 1982.

[Gho08] S. Ghosh. Strategies made explicit in dynamic game logic. In *Proceedings of the Workshop on Logic and Intelligent Interaction, ESSLLI 2008*, pages 74–81, 2008.

[Gor01] V. Goranko. Coalition games and alternating temporal logics. In *Proceedings of 8th conference on Theoretical Aspects of Rationality and Knowledge*, pages 259–272, 2001.

[Gor03] V. Goranko. The basic algebra of game equivalences. *Studia Logica*, 75(2):221–238, 2003.

[Grä08] E. Grädel. Banach-Mazur games on graphs. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 364–382, 2008.

[GS53] D. Gale and F. Stewart. Infinite games with perfect information. *Annals of Mathematics Studies*, 28:245–266, 1953.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, October 2002.

[HFMV95] J.Y. Halpern, R. Fagin, Y. Moses, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[Hin68] J. Hintikka. Language-games for quantifiers. In *Studies in Logical Theory*, volume 2 of *American Philosophical Quarterly Monograph Series*, pages 46–72. Blackwell, 1968.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, October 2000.

[HP78] D. Harel and V.R. Pratt. Nondeterminism in logics of programs. In *Proceedings of 5th Symposium on Principles of programming languages*, pages 203–213. ACM, 1978.

[HR87] J.Y. Halpern and M.O. Rabin. A logic to reason about likelihood. *Artificial Intelligence*, 32:379–405, 1987.

[HS87] J. C. Harsanyi and R. Selten. *A general theory of equilibrium selection in games*. MIT Press, 1987.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesly, 1979.

[HvdHMW03] P. Harrenstein, W. van der Hoek, J.J. Meyer, and C. Witteven. A modal characterization of Nash equilibrium. *Fundamenta Informaticae*, 57(2-4):281–321, 2003.

[Jon80] A. Jones. *Game Theory: Mathematical Models of Conflict*. John Wiley, 1980.

[JvdH04] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[Kec95] A. Kechris. *Classical descriptive set theory*. Springer, 1995.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KV99] O. Kupferman and M. Y. Vardi. Church's problem revisited. *The Bulletin of Symbolic Logic*, 5(2):245–263, 1999.

[KV01] O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *In proceedings of LICS'01*, pages 389–398. Computer Society Press, 2001.

[LR57] D.R. Luce and H. Raiffa. *Games and Decisions*. Wiley, 1957.

[Mad01] P. Madhusudan. *Control and synthesis of open reactive systems*. PhD thesis, Institute of Mathematical Sciences, November 2001.

[Mar75] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

[Mau81] R.D. Mauldin, editor. *The Scottish Book. Mathematics from the Scottish Café*. Birkhäuser, 1981.

[MS87] D.E. Muller and P.E. Shupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[Mul63] D.E. Muller. Infinite sequences and finite machines. In *Proceedings of the 4th IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 3–16, 1963.

[MW03] S. Mohalik and I. Walukiewicz. Distributed games. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, volume 2914 of *LNCS*, pages 338–351. Springer, 2003.

[Nas50] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:89–93, 1950.

[Ney85] A. Neyman. Bounded complexity justifies cooperation in the finitely repeated prisoner's dilemma. *Economic Letters*, 19:227–229, 1985.

[Ney98] A. Neyman. Finitely repeated games with finite automata. *Mathematics of Operations Research*, 23:513–552, 1998.

[OR94] M.J. Osborne and A. Rubinstein. *A course in game theory.* MIT Press, 1994.

[Par85] R. Parikh. The logic of games and its applications. *Annals of Discrete Mathematics*, 24:111–140, 1985.

[Pau01] M. Pauly. *Logic for Social Software.* PhD thesis, University of Amsterdam, October 2001.

[PP04] D. Perrin and J.E. Pin. *Infinite words: automata, semigroups, logic and games.* Academic Press, 2004.

[PR79] G.L. Peterson and J.H. Reif. Multi-person alternation. In *Proceedings of the 20th Symposium on Foundations of Computer Science*, pages 348–363. IEEE, 1979.

[PR90] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceeding of the 31st IEEE Symposium on Foundations of Computer Science*, pages 746–757, 1990.

[Pra76] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th Symposium on Foundations of Computer Science*, pages 109–121. IEEE, 1976.

[PRS09a] S. Paul, R. Ramanujam, and S. Simon. Dynamic restriction of choices: a preliminary logical report. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2009)*, pages 218–226, 2009.

[PRS09b] S. Paul, R. Ramanujam, and S. Simon. Stability under strategy switching. In *Proceedings of the 5th Conference on Computability in Europe, CiE 2009*, volume 5635 of *LNCS*, pages 389–398. Springer, 2009.

[Rab69] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Rei84] J.H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29:274–301, 1984.

[Sch04] P.Y. Schobbens. Alternating time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

[Sel65] R. Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeischrift Für Die Gesamte Staatswissenschaft*, 121:301–324, 1965.

[Str81] R.S. Streett. Propositional dynamic logic of looping and converse. In *Proceedings of 13th Symposium on Theory of computing*, pages 375–381, 1981.

[Str93] P.D. Straffin. *Game Theory and Strategy*. The Mathematical Association of America, 1993.

[Tel87] R. Telgársky. Topological games: On the 50th anniversary of the Banach-Mazur game. *Rocky Mountain J.Math.*, 17:227–276, 1987.

[Ten04] M. Tennenholtz. Program equilibrium. *Games and Economic Behaviour*, 49(2):363–373, 2004.

[Tho97] W. Thomas. Languages, automata, and logic. *Handbook of Formal Languages*, 3:389–455, 1997.

[Umm05] M. Ummels. Rational behaviour and strategy construction in infinite multi player games. Master's thesis, RWTH Aachen, 2005.

[vdHJW05] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 157–164, 2005.

[vdHW02] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*, pages 1167–1174, 2002.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, 1986.

[WvdHW07] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2007)*, pages 269–278, 2007.

[Zer13] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels,. In *Proceedings of the Fifth Congress Mathematicians*, pages 501–504. Cambridge University Press, 1913.