Reasoning *in* games

R. Ramanujam and Sunil Simon

The Institute of Mathematical Sciences
C.I.T. Campus, Chennai 600 113, India.
E-mail: {jam,sunils}@imsc.res.in

Abstract. We present a quick summary of formal reasoning about games in logics, pointing out that strategies (which are the core of game playing) are mostly treated as being atomic (unstructured). While this is adequate for reasoning "from above", where one is interested in the logical structure of the game itself, or that of removing dominated strategies, or existence of equilibria, it offers little help for how to play. When it comes to reasoning *within* the game, a player's choice of strategy depends not only on her knowledge of the game position but also her expectation of the strategies employed by other players of which she can only have partial knowledge. Moreover, strategies are completely known to players only in small games where a strategy amounts to picking a small number of moves; in games with rich structure, intricate reasoning about strategies can be called for. This article is intended as an (expository) account of only the main issues, with detailed proofs to be found in the referenced literature.

1 Introduction

A game can be thought of as any situation where there are multiple players, each having a number of *strategies* available which he can choose to follow, with a preference ordering on the possible outcomes, which in turn are determined by the strategies chosen by the players. Conflicts arise among players (who can be individuals or abstract entities) when they attach different values to an outcome. Players might be in direct conflict with each other or a group of players might try and cooperate to achieve a common goal.

This abstract definition turns out to be very robust and enables social interactions from a wide range of fields to be theoretically modelled as a game. A few examples include companies adopting marketing strategies to win over customers, candidates adopting campaigning plans in an election, biological species trying to pass their respective genes to future generations. A comprehensive collection of problems from various fields being modelled as games can be found in [Str93].

Game theory tries to analyse situations where there are elements of conflict and cooperation among rational agents. The ultimate aim of the theory is to predict the behaviour of rational agents and prescribe a plan of action that needs to be adopted when faced with a strategic decision making situation. The theory therefore consists of the modelling part as well as the various solution concepts which try to predict the behaviour of such agents and prescribe what rational players should do. The effectiveness of a particular solution concept depends on how precise and effective the prescription turns out to be.

There are some fundamental problems in coming up with such a theory. Most real world interactions are extremely complex and modelling the complete process as a game may not be feasible. The usual approach to overcome this trouble is to consider an abstraction of the situation and to model this abstract setting as a game. Even though all the constituent elements of the situation cannot be preserved, the abstraction tries to retain the most relevant ones. The other challenge is due to the fact that game theory assumes that all players are *rational* whereas in many real world interactions, people often tend to act irrationally.

Games can be represented in various ways depending on the amount of strategic details that need to be preserved during the abstraction process. The two popular methods of representation are the strategic form (or normal form) which results in "small" games and the extensive form used to represent "large" games where the game structure is presented explicitly.[1]

This article, which is mainly an expository account, is organised as follows. We begin with strategic form games and then outline two approaches to reason about large, complex games. One works with algebraic structure in a large game, and the other with graphical structure. We then try and motivate the necessity to reason about strategies rather than be content with reasoning about games. We present a syntax for strategy specifications (similar to the alegebraic structure for games) to describe partially known strategies on graphical games, and show how notions like equilibria may be studied in such a setting.

Strategic form games

Consider the classic situation of two children wanting to divide a piece of cake among themselves. The solution is to let one child divide the cake and let the other choose which piece he wants. Each child wants to maximise the size of his piece, and therefore this process ensures fair division. The first child cannot complain that the cake was divided unevenly and the second child cannot object since he has the piece of his choice. This is a very simple example of a game where two players have conflicting interests and each player is trying to maximize his payoff. The final outcome of the division depends on how well each child can anticipate the reaction of the other and this makes the situation game theoretic.

A game can be presented by specifying the players, the strategies available to each player and the payoff for each player. In the case of a two person game, this can be presented efficiently in a matrix form. For instance the cake cutting game can be represented using the matrix shown in Figure 1. We will refer to the players as *cutter* and *chooser*. Here both players have two strategies, each can choose to cut the cake evenly or to make one piece bigger than the other, which corresponds to picking one of the rows of the matrix. Chooser can choose the bigger piece or the smaller piece, which corresponds to picking one of the columns of the matrix. The outcome for the cutter, after both the players choose their strategies, is the corresponding entry in the matrix. For instance, if the cutter chooses to make one piece bigger and the chooser picks the bigger piece, then the outcome will be that the smaller piece goes to the cutter (bottom left cell). The chooser's outcome is the complement of the cutter's. An equivalent representation of the game can be obtained by replacing the outcomes with numbers representing *payoffs* as shown in Figure 2.

	Choose bigger piece	Choose smaller piece
Cut the cake evenly	Half of the cake	Half of the cake
Make one piece bigger	Small piece	Big piece

Fig. 1.

[1] Note that "small" and "large" are informal notions relating to the typical number of moves in such games.

	Choose bigger piece	Choose smaller piece
Cut the cake evenly	0	0
Make one piece bigger	-1	1

Fig. 2.

The cake cutting game captures the situation of pure conflict, where cutter's gain is chooser's loss and vice-versa. Such games are called *zero-sum* or *win-loss* games. If the cutter had the option of choosing any of the four available outcomes, he would prefer to have the big piece. However, he realizes that expecting this outcome is highly unrealistic. He knows that if he were to make one piece bigger, then the chooser will pick the bigger piece leaving him with the remaining smaller one. If he divides evenly, then he will end up with half of the cake. The cutter's choice is really between the smaller piece and half of the cake. Therefore he will choose to take half of the cake (top left cell) by making an even split of the cake. This amount is the maximum row minimum and is referred to as the *maximin*.

Now consider a variation of this game where the chooser is required to announce his choice (big or small piece) before the cake is cut. This does not change the situation, the chooser would still choose a bigger piece irrespective of how the cutter divides the cake. i.e. the chooser looks for the minimum column maximum (*minimax*) value, which happens to be the top left cell. In this example, the maximin value and the minimax value both happen to be the top left cell. For a game when the maximin value and the minimax value is identical, the outcome is called the *saddle point*. When a game has a saddle point it is the expected rational play since either player cannot unilaterally improve his payoff. A win-loss game is said to be *determined* if a saddle point exists.

Formally a two player zero sum game where player 1 has m strategies and player 2 has n strategies can be represented by an $m \times n$ array A, where the (i, j)th entry $a_{i,j}$ represents the payoff of player 1 when he chooses the strategy i and player 2 picks strategy j. The payoff for player 2 for the corresponding entry is $-1 \times a_{i,j}$. Note that non zero sum payoffs can easily be represented by replacing each matrix entry by a tuple of payoffs for each player.

	Heads	Tails
Heads	1	-1
Tails	-1	1

Fig. 3. Matching pennies

Unfortunately not all games have saddle points. One of the simplest examples is the game of "Matching pennies" depicted in Figure 3. In this game two players simultaneously place a penny (heads or tails up). When both the pennies match, player 1 gets to keep both. If the pennies do not match, then player 2 gets to keep both. Its easy to see from the payoff matrix that maximin is -1 whereas minimax is 1. It is well known that the best way of playing matching pennies is to play heads with probability half and tails with probability half. This amounts to a *mixed strategy* rather

than a *pure strategy* of picking an action with absolute certainty. The minimax theorem [vNM47] asserts that for all two player zero sum games, there is a rational outcome in mixed strategies.

The theory can be extended from zero sum objectives to non zero sum objectives with more than just two players. In this case the outcome of the game will specify a payoff for each of the players. The commonly used solution concept in this context is that of Nash equilibrium which corresponds to a profile of strategies, one for each player which satisfies the property that no player gains by unilaterally deviating from his equilibrium strategy. John Nash [Nas50] formulated this notion of equilibrium for multiplayer non zero sum games and proved the analogue of the min-max theorem for such games. The result states that for all finite multiplayer games, there exists a mixed strategy (Nash) equilibrium profile.

Much of the mathematical theory developed for games talks about existence of equilibrium and does not shed light on how the players should go about playing the game. For two person zero sum games, one can show that the maximin theorem is equivalent to the LP (linear programming) duality problem. Therefore construction of optimal strategies is possible using linear programming techniques [vS02]. For two person non zero-sum games, optimal strategies can be constructed using techniques for solving the linear complementarity problem as shown in [LJ64]. For a multi-player game, Nash's theorem talks of existence of equilibrium but it is not known how to actually construct the equilibrium strategy.

2 Reasoning about games

Strategic form games give a highly abstracted presentation of a game. The representation typically assumes "small" games where the structure of the strategy (individual moves which build up to form the strategy) is absent (or abstracted away). In the context of pure strategies, the above mentioned existence theorems dictate which strategy a player should employ in the game. However, we also need to analyse larger games where the players' actions are part of the representation. We now address reasoning in such a context.

2.1 Game logics

One natural way is to consider a large game as being built up structurally from small atomic games by means of composition. This suggests an algebraic structure in games, and one line of work in game logics proceeds by imposing a program-like compositional structure on games.

Program logics like the propositional dynamic logic (PDL) [HKT00] have been developed to reason about programs. The idea here is to model programs as being constructed using operations like sequential composition, iteration, etc. on simple atomic programs. This compositional approach in program reasoning has been successful in the analysis and verification of programs, especially in giving us insights into the expressive power of various programming constructs. The natural extension to this methodology is to come up with a dynamic logic to reason about multi-agent programs and protocols. Game logic introduced by Parikh in [Par85] addresses this issue. Game logic (GL) is a generalisation of PDL for reasoning about determined two person games.

Syntax: Let the two players be denoted as player 1 and player 2. Like PDL, the language of GL 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 GL-games Γ and the set of GL-formulas Φ is built from the following syntax:

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

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 fact that player 2 has a strategy to ensure φ. The intuitive definition of composite games are 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 same as playing the game γ with the roles interchanged. 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:

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

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 $E_\gamma(Y) = \{s \in S \mid Y \in E_\gamma(s)\}$. Then

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

where μ denotes the least fixpoint operator. It can be shown that the monotonicity of E_g is preserved under the game operations and therefore the least fixpoint $\mu X.Y \cup E_\gamma(X)$ always exists.

Since game logic was designed to reason about multi-agent programs, the modelling approach is quite different from traditional game theoretic notions. Pauly in [Pau00] presents a semantics for Game logic which is closer to the standard game-theoretic approach.

Satisfiability and model checking are fundamental decision problems associated with any logic. For Game logic, the following theorems show that both are decidable.

Theorem 2.1. *([Par85]) The satisfiability problem for Game Logic is in EXPTIME.*

Theorem 2.2. *Given a Game Logic formula φ and a finite game model M, model checking can be done in time $\mathcal{O}(|M|^{ad(\varphi)+1} \cdot |\varphi|)$ where $ad(\varphi)$ is the alternation depth of φ.*

A proof of this theorem can be found in [Pau01], Theorem 6.21 (page 122).

As shown in [Par85], it is possible to interpret Game logic over Kripke structures. Over Kripke structures, Game logic can be embedded into μ-calculus [Koz83]. Whether Game logic is a proper fragment of the μ-calculus is not known. It is quite conceivable that model checking for Game logic is easier than model checking for the full μ-calculus. However, Berwanger in [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 that it is

complete, unfortunately no proof of this has been given so far. For the dual free fragment of Game logic, a complete axiomatization is presented in [Par85].

In Game logic, starting with simple atomic games, one can construct large complex games using operators like composition and union. Due to the presence of the Box-Diamond duality $\langle\gamma\rangle\varphi \equiv \neg[\gamma]\neg\varphi$, it is easy to see that 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 emphasis in this approach is to reason about games, to study the structure of games with interesting properties and definability conditions.

2.2 Graphical games

Graphs are another convenient way of representing large games in such a way that the structure of the game is preserved. The nodes of the graph are game positions and edges represent player moves that are enabled. In this article we consider only *turn based games*: these are games where at any position a single player moves. This can be modelled by having every game position (except for leaf nodes) being assigned to a particular player. A play is then just a path (a sequence of vertex and edge labels) in the graph. Traditional games like chess, bridge, backgammon etc. have a set of rules associated with them which specifies the legal moves of each player. Such games can be easily transformed into graph games where the game positions and actions are derived from the rules specified. A graphical game therefore consists of a game arena (the game graph) along with a winning condition. Formally we can define it as follows:

Game Arena: Let $\Lambda = \{1, 2, \ldots, n\}$ be a finite set of players and $\Sigma = \{a_1, a_2, \ldots, a_m\}$ be a finite set of action symbols, which represent moves of players. For a graph $G = (W, \longrightarrow)$ with vertex set W and edge relation $\longrightarrow: W \times \Sigma \to W$, let the set of successors of $s \in W$ be defined as $\vec{s} = \{s' \in W \mid s \xrightarrow{a} s' \text{ for some } a \in \Sigma\}$. A node $s \in W$ is a *terminal* node if $\vec{s} = \emptyset$. A **game arena** is a graph $\mathcal{G} = (W, \longrightarrow, s_0)$ with $W = \bigcup_{i\in\Lambda} W^i \cup \{W^{leaf}\}$. For $i \in \Lambda$, W^i is the set of *game positions* for player i and W^{leaf} is the set of terminal game positions. s_0 is the initial node of the game and the transition function $\longrightarrow: (W \times \Sigma) \to W$ is a partial function also called the move function. In an arena, the play of a game can be viewed as placing a token on s_0. If player i owns the game position s_0 (i.e $s_0 \in W^i$), then she picks an action 'a' which is enabled for her at s_0 and moves the token to s' where $s_0 \xrightarrow{a} s'$. The game then continues from s'. Formally, a play in $\mathcal{G}$ is a (possibly infinite) path $\rho : s_0 a_0 s_1 a_1 \cdots$ where $\forall j : s_j \xrightarrow{a_j} s_{j+1}$. Let *Plays* denote the set of all plays in the arena.

The arena $\mathcal{G}$ as defined above merely describes the rules by which the game progresses. More interesting are the objectives of the players, which specify the game outcomes. We assume that each player has a preference relation over the set of plays. Let $\preceq^i \subseteq (Plays \times Plays)$ be a complete, reflexive, transitive binary relation denoting the preference relation of player i for $i \in \Lambda$. Then the game G is given as, $G = (\mathcal{G}, \{\preceq^i\}_{i\in\Lambda})$.

Determinacy for reachability games: We first consider the case of games (on possibly infinite graphs) where there are two players (i.e. $\Lambda = \{1, 2\}$) with strictly conflicting objectives. Win-loss objectives can be presented in a very simple way by just specifying the set of winning plays for a player (say player 1) instead of giving the explicit preference relation for both the players. The simplest winning condition is the reachability condition which specifies a play to be winning for player 1 if a game position in a specific set of good game positions is reached at least once.

The first step in analysing such a game is to check if determinacy holds. Unlike in Game Logic, where determinacy can be presented as a formula, showing determinacy directly using game-theoretic techniques is quite non-trivial. Determinacy for reachability games however can be shown quite easily using a technique similar to backward induction [Kuh53] on extensive form game trees. The idea is to inductively build the set of game positions U from which player 1 has a winning strategy. Initially we take U to be the set of all good game positions specified by the reachability condition and at any stage expand this set by its *attractor set* [Zie98] which is basically the set of all game positions from where player 1 can force a visit into some vertex of U in finitely many steps.

Determinacy for regular objectives: When we consider **infinite plays**, more interesting general winning conditions (than reachability) can be considered. For instance, a infinite play is said to be winning for player 1 if a specific set of game positions is visited during the play *infinitely often*. This is referred to as a Büchi condition. Alternatively, several such sets may be given and we may insist that the set of all infinitely often visited positions in a winning play should figure among the given ones. This is called the Muller condition. Such conditions broadly come under the category of ω-regular objectives ([GTW02]). Parity objectives, whereby we associate a finite set of numbers to label game positions and determine winning by considering the parity of (say) the highest visited number, neatly generalize the abstractions studied in this manner.

For games with ω-regular objectives, determinacy follows from a very general result due to Martin [Mar75] which states that all Borel games (a class containing regular objectives) are determined. Adopting this technique however results in winning strategies which in general need to keep track of the entire history of the play and therefore not implementable by finite state machines. Of particular interest are the following two types of strategies:

- Positional (memoryless) strategies: These are strategies for which the next move depends only on the current game position and not on the entire history of play.
- Finite 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.

Both positional and finite memory strategies can be implemented using finite state automata where the state space of the automaton corresponds to the memory required by the strategy. In particular a positional strategy can be implemented by an automaton with a single state.

For a game with regular winning condition given by the Muller condition, Gurevich and Harrington [GH82] show that finite memory strategies suffice for winning. In the case of parity games, the existence of memoryless winning strategy was shown by Emerson and Jutla [EJ91].

Applications to automata theory: Establishing positional or finite memory determinacy for a class of games is only a first step in the analysis of those games. For the determinacy theorem to be useful, it is also crucial that we are able to construct the winning strategy. For games played on a finite game arena, we can strengthen the determinacy result from an existence result to one where the winning strategy can be effectively constructed. The *Büchi-Landweber theorem* [BL69] states that for games played on finite graphs the set of "winning" game positions for each of the players, along with the respective winning strategy can be effectively computed. McNaughton [McN93] presents a simpler constructive proof for Muller games played on finite game arena where he gives an exponential time algorithm to compute the finite memory winning strategy. Since finite memory strategies can be implemented using finite state automata, the above mentioned theorems imply that for two player zero sum games played on a finite game arena with an ω-regular winning condition, the game is determined and the winning strategy can be synthesized as a finite state automaton.

The techniques developed in the analysis of infinite two player zero sum graphical games have been used extensively in areas outside game theory especially in automata theory and to show the decidability of various modal logics and monadic second order logics. For instance, it has been shown that the model checking problem for μ-calculus is equivalent to deciding the winner in a parity game via a linear time reduction [EJS93,Wil01]. Along with the result in [Ber03] (mentioned in section 2.1) this also implies that model checking Game logic is not easier than solving parity games. The closure under complementation of automata on infinite trees is another consequence of the determinacy theorem for parity games as shown in [GH82]. This gives an elegant proof of the difficult part of Rabin's theorem [Rab69] which shows the decidability of monadic second-order theory of the infinite binary tree.

A detailed exposition on the applications of techniques developed for infinite games in automata theory and logic can be found in [GTW02].

Overlapping objectives: Infinite games on graphs have been extensively studied in the two player zero sum setting. It makes sense to also look at multi-player non zero-sum games on graphs. Since objectives need not be strictly conflicting, each player can have a preference relation inducing an ordering over the set of valid plays. The game is specified by presenting the game arena along with the preference relation for each player. The notion of equilibrium remains the same as seen in section 1. However, it is no longer clear how to construct an equilibrium strategy profile for a game in general. For games where the objectives of each player is specified by an omega regular condition, the existence of Nash equilibrium is shown in [CJM04]. Ummels in [Umm06] shows the existence of sub game perfect equilibrium for such games.

There is also the question of rational behaviour when we consider games which are not strictly competitive. In the case of finite games, the assumption of rational behaviour has led to the classical concept of *iterated admissibility*, which is based on the well studied notion of weak dominance of strategies. A strategy μ_1 of a player is said to dominate strategy μ_2 if against any choice of strategies of the other players, μ_1 performs at least as well as μ_2 and there are cases when μ_1 performs strictly better than μ_2. As a consequence of rationality, when a player takes into account all strategies of the other players, she will avoid playing dominated strategies. Each player eliminates from her set of strategies those which are dominated. Since rationality is common knowledge each player knows which strategies his opponents eliminated, and as a result he might discover that some of his remaining strategies are dominated when taking into account the new set of possible strategies by the opponents. This leads to further elimination. For games with finite strategy spaces it is easy to see that this process stabilises after a finite number of iterations. A solution concept like the Nash equilibrium predicting the outcome of the game should only involve strategies that survive iterated elimination since those are the only strategies a rational player will need to reason about.

For infinite non zero-sum multi-player games, since it is not clear how to come up with an equilibrium strategy profile, we need some methods to prune down the strategy space that needs to be considered for equilibrium outcomes. This can be achieved by generalising the concept of iterated admissibility to infinite games, since any equilibrium strategy is required to be iteratively admissible. However, the generalisation is not very obvious as the elimination process might not even terminate. Berwanger in [Ber07] studies the question of rational behaviour in this context, and shows that in the case of ω-regular games, the iteration always terminates after a finite number of steps and that the set of iteratively admissible strategies is itself regular.

3 Reasoning in games

The pattern of reasoning about games presented in the previous section consists of trying to find when games are determined and characterising the type of winning strategy that needs to be

employed. Throughout the discussion so far, strategies are assumed to be complete plans which prescribe a unique action for every possible scenario. Therefore coming up with a winning strategy requires the entire game graph to be explored and analysed. But this may be a tall order: even when the game is finite, the state space could be so large that exploring all possibilities might be computationally infeasible.

A classic example of such a game is the game of chess. Zermelo showed in [Zer13] 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. For games like Hex, it is known that the first player can force a win [Gal79] but nonetheless a winning strategy is not known. Connect four is yet another game which has been shown to be determined [All98]. Theoretically a finite game like chess or hex is not very interesting since in principle, the winning strategy can be found out in a finite amount of time. The winning strategy is positional and therefore can be presented as a finite state automaton as well.

Unfortunately these results are of no help in advising a player on how to play in an actual game. The situation gets worse in the case of games with overlapping objectives. In general, such games can have multiple Nash equilibria and even if we forget about the computational issues, it still leaves the problem of which equilibrium the players should try to attain. Harsanyi and Selten in [HS87] develop an equilibrium selection theory where a unique Nash equilibrium is selected. However, due to the complexity associated with the theory, it is not clear if players would actually employ this in order get advice on how to play. In such situations rather than be content with reasoning *about games* using the functional notion of strategies, one needs to reason *about strategies*. Strategies need to be compared with each other to check which ensures better outcome, and their structure needs to be explicitly analysed. Moreover the strategy employed by a player may well depend on her knowledge of or belief regarding the strategies followed by other players.

This is a departure from the classical solution concepts in game theory where the emphasis is on figuring out what happens if a player deviates from the equilibrium strategy. Since specifying the complete strategy may not be possible, one needs to look at partial specifications of strategies and try to compose such specifications in some manner. Heuristics for instance, are extensively used in many of the chess playing programs [CR,GC]. Listed below are some of them [Lev90,RN02]:

- Alpha-beta pruning: This is employed to reduce the number of possibilities evaluated in a search tree by the min-max procedure. It stops evaluating a move completely when it finds the existence of at least one witness which shows that the move is inferior to earlier examined ones.
- The killer move heuristic: Alpha-beta pruning is most efficient when the best moves are considered first. The killer move heuristics tries to guess the best move. The idea being that a move which is good from a different but similar position might also be a good move in the current position.
- Quiescence search: Due to the computational restrictions there is a limit to the number of moves that a machine can search ahead. Humans on the other hand are good at pattern matching and might be able to identify board positions which are interesting and search deeper only on a particular path. Quiescence search tries to emulate this behaviour by making the computer search "interesting" positions to a greater depth than the other positions.

It is quite evident that these heuristics are basically partially specified strategies. A library of such specifications is developed and during the course of play, the actual strategy is built up by composing various partial strategies. Checkers is another game where strong heuristics based game playing programs are available. By performing extensive strategy analysis using computers it was recently shown that perfect play in checkers will result in a draw [SBB+07].

Aumann and Dreze [AD05] make a strong case for the focus of game theory to shift from equilibrium computation to questions of how rational players should play. For zero sum games, the value of the game is unique and rational players will play to achieve this value. However, in the case of non zero sum games as mentioned above, multiple Nash equilibria can exist. This implies that players cannot extract an advice as to which strategy to employ from the equilibrium values. According to Aumann and Dreze, for a game to be well defined, it is also necessary that players have an expectation on what the other players will do. In estimating how the others will play, a rational player should take into account that others are estimating how he will play. 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. The strategy specifications we introduce below are in the same spirit, since such a specification will be interactive in the sense of [AD05].

We look at multi-player non-zero sum graphical games played on finite game arena. Our approach is to combine the techniques presented above to do reasoning *in* games rather than reason about games. We propose a syntax for building strategies, thereby using the techniques developed for graphical games and the ideas in Game logics for reasoning about the strategies in games rather than reasoning about the games themselves. In this work, the emphasis is on coming up with techniques which can advise the players on how to go about playing the game and not on showing existence results that game theory is well known for. Since this requires that we explicitly analyse the structure of the game, graphical games are natural models for our purpose.

In [vB01] van Benthem uses dynamic logic to describe games as well as strategies and in [HvdHMW03] and [vdHJW05], van der Hoek and co-authors develop logics for strategic reasoning and equilibrium concepts. This work is close to ours in spirit; however our point of departure is in bringing logical structure into strategies rather than treating strategies as atomic.

We use the graphical game models as defined in section 2.2. For ease of presentation, the technical machinery is developed only for two player games, but the analysis carries over to n-player games where $n > 2$. For convenience we also restrict our attention to finite plays. For this purpose we assume that Σ has a special *exit* action and that the game arena has a unique terminal node denoted by *leaf*. For an arena $\mathcal{G} = (W, \longrightarrow, s_0)$, we assume that the transition function satisfies the following property:

- For all $s, s' \in W$ such that $s' \xrightarrow{x} s'$, $s' = \mathit{leaf}$ iff $x = \mathit{exit}$.

Let $\overline{i} = 2$ when $i = 1$ and $\overline{i} = 1$ when $i = 2$.

Strategies and equilibrium

Let $\mathcal{G}_T$ denote the tree unfolding of the arena $\mathcal{G}$. A strategy for player 1, $\mu = (W_\mu, \longrightarrow_\mu, s_0)$ is a maximal connected subtree of $\mathcal{G}_T$ where for each player 1 node, there is a unique outgoing edge and for the other player every move is included. That is, for $s \in W_\mu$ the edge relation satisfies the following property:

- if $s \in W^1_\mu$ then there exists a unique $a \in \Sigma$ such that $s \xrightarrow{a}_\mu s'$, where we have $s \xrightarrow{a}_T s'$.
- if $s \in W^2_\mu$, then for each s' such that $s \xrightarrow{a}_T s'$, we have $s \xrightarrow{a}_\mu s'$.

Let Ω^i denote the set of all strategies of Player i in $\mathcal{G}$, for $i = 1, 2$. A strategy profile $\langle \mu, \tau \rangle$ defines a unique path ρ^τ_μ in the game $\mathcal{G}$. This path constitutes a valid *play* if it is of the form $s_0 a_0 \cdots a_{n-1} s_n$ where $s_n = \mathit{leaf}$. The notion of best response and equilibrium can be defined as follows.

- μ is the best response for τ iff ρ^τ_μ constitutes a valid play and $\forall \mu' \in \Omega_1$ such that $\rho^\tau_{\mu'}$ is a play we have $\rho^\tau_{\mu'} \preceq^1 \rho^\tau_\mu$.

– Symmetric definition for τ.
– $\langle \mu, \tau \rangle$ is a Nash equilibrium iff μ is the best response for τ and τ is the best response for μ.

The natural algorithmic questions that are of interest include:

– Given a strategy τ of player 2, what is the best response for player 1?
– Given a strategy profile $\langle \mu, \tau \rangle$, is it a Nash equilibrium?
– Does the game possess a Nash equilibrium?

Clearly, if we can answer the first question, we can answer the second as well. In any case, to study these questions algorithmically, we need to be able to present the preferences of players' and their strategies in a finite fashion. We do this by presenting the strategies and preference relations of players using finite state automata as shown below.

Advice Automata: In this article, we restrict our attention to *bounded memory* strategies, which can be represented using finite state automata. We think of these as advice automata, in the sense that they constitute an advice for the player to consult at a node. Since our objective is to work with strategies specified by their properties, it makes sense to see them as automata which "accept" a set of strategies.

For a game graph $\mathcal{G}$, a nondeterministic advice automaton for player i is a tuple $\mathcal{A} = (Q, \delta, o, I)$ where Q is the set of states, $I \subseteq Q$ is the set of initial states, $\delta : Q \times W \times \Sigma \to 2^Q$ is the transition relation, and $o : Q \times W^i \to \Sigma$, is the output or advice function.

The language accepted by the automaton is a set of strategies of player i. Given a strategy $\mu = (W_\mu, \longrightarrow_\mu, s_0)$ of player i, a run of $\mathcal{A}$ on μ is a Q labelled tree $T = (W_\mu, \longrightarrow_\mu, \lambda)$, where λ maps each tree node to a state in Q as follows: $\lambda(s_0) \in I$, and for any s_k where $s_k \xrightarrow{a}_\mu s'_k$, we have $\lambda(s'_k) \in \delta(\lambda(s_k), s_k, a_k)$.

A Q labelled tree T is accepted by $\mathcal{A}$ if for every tree node $s \in W^i_\mu$, if $s \xrightarrow{a}_T s'$ then $o(\lambda(s)) = a$. A strategy μ is accepted by $\mathcal{A}$ if there exists an accepting run of $\mathcal{A}$ on μ. It is easy to see that any bounded memory strategy can be represented using a *deterministic* advice automaton.

Evaluation Automata: The preferences of players can also be presented as a finite state automaton which runs over plays. Let B be a finite state automaton with final states being F. We fix a set of colours C with a preference relation $\lhd^i$ for each player over this set along with a function which maps a final state to each colour in C. The preference relation induces a natural preference ordering over plays by considering the final state reached at the end of the play (note that all plays are finite by definition). The purpose of the evaluation automaton running on a game arena $\mathcal{G}$ can be thought of as checking for certain properties of paths in the arena and providing an appropriate classification. We will use $\mathcal{E}$ to denote an evaluation automaton. Given $\mathcal{E}$ with the set of colours C, for each player i we can always build a set of classical win-loss evaluation automata, one for each colour $c \in C$ where the new automaton wins on all colours at least as preffered as c. We will use $\mathcal{E}^i_c$ to denote such win-loss automata.

4 Strategy specification

We now give a syntax to specify strategies in a structured manner. The atomic case specifies, for a player, what conditions she tests for before making a move. We can associate with the game graph 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 $Past(X)$ be sets of formulas given by the following syntax:

$$\psi \in Past(X) := x \in X \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \diamondsuit\psi.$$

Syntax

Let $P^i = \{p_0^i, p_1^i, \ldots\}$ be a countable set of observables for $i \in \{1, 2\}$ and let $P = P^1 \cup P^2$.

The syntax is given by:

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

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

Semantics

Given any sequence $\xi = t_0 t_1 \cdots t_m$, $V : \{t_0, \cdots, t_m\} \to 2^X$, and 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 $\xi, k \models \psi_2$.
- $\xi, k \models \diamondsuit\psi$ iff there exists a $j : 0 \leq j \leq k$ such that $\xi, j \models \psi$.

We consider the game arena $\mathcal{G}$ along with a valuation function for the observables $V : W \to 2^P$. Given a strategy μ of player i and a node $s \in \mu$, let $\rho_s : s_0 a_0 s_1 \cdots s_m = s$ be the unique path in μ from the root node to s. For a strategy specification $\sigma \in Strat^i(P^i)$, we define when μ conforms to σ (denoted $\mu \models_i \sigma$) as follows:

- $\mu \models_i \sigma$ iff for all player i nodes $s \in \mu$, we have $\rho_s, s \models_i \sigma$.

where we define $\rho_s, s_j \models_i \sigma$ for any s_j in ρ_s as,

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

Above, $\pi \in Strat^{\overline{i}}(P^1 \cap P^2)$, $\psi \in Past(P^i)$, and for all $i : 0 \leq i < m$, $out_{\rho_s}(s_i) = a_i$ and $out_{\rho_s}(s)$ is the unique outgoing edge in μ at s.

The following lemma relates structured strategy specifications to advice automata.

Lemma 4.1. *Given a player $i \in \{1, 2\}$ and a strategy specification σ, we can construct an advice automaton $\mathcal{A}_\sigma$ such that $\mu \in Lang(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.*

Proof. We proceed by induction on the structure of σ. We construct automata for atomic strategies and compose them for complex strategies. The states of the automaton consists of sets of subformulas of the past time formulas appearing in σ. Note that the strategy is implemented principally by the output function of the advice automaton.

($\sigma \equiv [\psi \mapsto a]$): The automaton works as follows. The automaton keeps track of past formulas satisfied along a play as game positions are traversed and that the valuation respects the constraints generated for satisfying ψ. The automaton also guesses a move at every step and checks that this is indeed a when ψ holds; in such a case this is the output of the automaton.

($\sigma \equiv \sigma_1 \cdot \sigma_2$): By induction hypothesis there exists $\mathcal{A}_{\sigma_1} = (Q_{\sigma_1}, \delta_{\sigma_1}, o_{\sigma_1}, I_{\sigma_1})$ 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 $\mathcal{A}_{\sigma_2}$.
($\sigma \equiv \sigma_1 + \sigma_2$): We take $\mathcal{A}_\sigma$ to be the disjoint union of $\mathcal{A}_{\sigma_1}$ and $\mathcal{A}_{\sigma_2}$. Since the automaton is nondeterministic with multiple initial states, we retain the intial states of both $\mathcal{A}_{\sigma_1}$ and $\mathcal{A}_{\sigma_2}$. If a run starts in an initial state of $\mathcal{A}_{\sigma_1}$ then it will never cross over into the state space of $\mathcal{A}_{\sigma_2}$ and vice versa.
($\sigma \equiv \pi \Rightarrow \sigma'$): By induction hypothesis there exists $\mathcal{A}_\pi = (Q_\pi, \delta_\pi, o_\pi, I_\pi)$ which accepts all player 2 strategies satisfying π and $\mathcal{A}_{\sigma'} = (Q_{\sigma'}, \delta_{\sigma'}, o_{\sigma'}, I_{\sigma'})$ which accepts all player 1 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. Till π is violated, it is forced to follow the transitions of $\mathcal{A}_{\sigma'}$.

For a strategy specification σ, the size of the advice automaton $\mathcal{A}_\sigma$ constructed using the above procedure will be exponential in the size of σ.

5 Comparing strategy specifications

A strategy specification denotes a set of strategies satisfying certain propeties rather than a single strategy and therefore the notion of strategy comparison needs to be re-examined in the new setting. Let C be the set of colours used to classify the runs appropriately by the evaluation automaton and $\lhd^i$ the preference ordering of player i. For a player $\overline{i}$ strategy specification π and player i specifications σ and σ', we have various possible definitions as to when σ is better than σ'. Given below are just a few of the possible definitions:

- σ is better than σ' if as long as player $\overline{i}$ conforms to π, if there is a $c' \in C$ and a strategy for player i which conforms to σ' ensuring condition c', then there also exists a $c \in C$ where $c' \lhd^i c$ and a strategy for player i which conforms to σ ensuring condition c.
- σ is better than σ' if as long as player $\overline{i}$ conforms to π, if for all strategies of player i, which conforms to σ' there is a $c' \in C$ such that the strategy ensures c' then for all strategies which conform to σ, there exists $c \in C$ such that the strategy ensures c where $c' \lhd^i c$.

Having chosen an appropriate notion for comparison, 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 specification pair (σ, π) constitutes an 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 $\mathcal{E}^i_c$ a win-loss evaluation automaton for player i.

- Does player i have a strategy conforming to σ which ensures a valid play that is winning for i with respect to $\mathcal{E}^i_c$, as long as player $\overline{i}$ is playing a strategy conforming to π (abbreviated as $\exists\sigma, \forall\pi : \mathcal{E}^i_c$)?
- 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 $\mathcal{E}^i_c$ (abbreviated as $\forall\sigma, \forall\pi : \mathcal{E}^i_c$)?

We call this the *verification* question. The *synthesis* question is given π and $\mathcal{E}^i_c$ to construct a strategy as a deterministic advice automaton $\mathcal{A}^i$ such that $\mathcal{A}^i, \forall\pi : \mathcal{E}^i_c$ holds (since $\mathcal{A}^i$ is deterministic, we can avoid the quantification).

Theorem 5.1. *([RS07]) Over finite game graphs with Muller objectives,*

- *The verification problem is decidable.*
- *The synthesis question is solvable.*

These results also enable us to check for a specification σ if it is the best response to the opponent's specification π and to synthesise an advice automaton which is the best response to the opponent's specification π.

A simple logic to reason about structured strategies is presented in [RS06]. The two main constructs of the logic are:

- $(\sigma)_i : a$ which says that the action a is enabled by the specification σ for player i at a game position.
- $\sigma \rightsquigarrow_i \beta$ which asserts that player i can play according to σ and "ensure" β. In other words, for an i node, there exists a choice of action enabled by σ which achieves β. For an $\overline{i}$ node, all actions result in β.

The construct $\sigma \rightsquigarrow_i \beta$ formalises in a logical framework the backward induction technique. In [RS06] a complete axiomatization of this logic is presented as well. The truth checking problem for this logic can also be shown to be decidable [RS07].

6 Example

Probably the best way to illustrate the notion of strategy specification is to look at heuristics used in large games like chess, go, checkers, etc. A heuristic strategy is basically a partial specification, since it involves checking local properties like patterns on the board and specifying actions when certain conditions hold. Heuristics are usually employed when the game graph being analysed is too large for a functional strategy to be specified. Below, we look at a few heuristics applied in chess. Instead of writing explicit formulas we employ a verbal description for convenience. However, it will be clear that the corresponding strategy specifications can be formally presented.

Chess

Fork: Fork or *double attack* are moves that attack two enemy targets simultaneously. Double attacks can be brought about mostly by knight, queen, bishop and pawn. The joy of attacking by pawn is that any two pieces (even if they are defended) are fine since pawn is the least valuable. Checking if a double attack by a pawn is possible is a local check, one just needs to look at the current game position. We can consider this to be an atomic proposition. Therefore we can have a specification as follows.

- If a pawn double attack is possible then play the action resulting in the fork.

Notice that we did not specify a condition of the form "if a pawn is on f2 and the opponent rook and knight are on e5 and g5 respectively then move f2-f4". This would constitute a specific advice which will be part of a functional strategy and not a generic one reflecting the abstract properties of states. In particular this functional strategy would conform to our generic specification mentioned above.

A very common forking piece is the knight because of its unique pattern of movement. The knight is roughly comparable in value to a bishop and is less valuable than a rook or a queen. We can effectively implement a double attack by knight if the attacked pieces are undefended or more valuable (even if it is defended). Therefore the specification will now have an extra conjunct to check these conditions.

– If a knight double attack is available and (the target pieces are not guarded or more valuable) then play the action resulting in the fork.

A queen can also bring about a double attack due to its almost unrestricted movement ability. A queen may sometimes be sacrificed for a checkmate, however it is not usually worthwhile to attack any defended piece with a queen. Therefore a queen double attack is usually implemented only if both target pieces are unguarded or if one target piece is unguarded and the other is the king. We can modify the specification as follows.

– If a queen double attack is available and both target pieces are not guarded then play the action resulting in the fork.
– If a queen double attack is available and (one target piece is not guarded and the other is the king) then play the action resulting in the fork.

The forking strategy itself can be expressed as a specification. For instance a property required to successfully fork with a knight is that the knight and all its targets are placed on squares of the same colour before the move (this is because any time a knight moves, it ends up in a different coloured square). One defence against double attack is to defend the "forking square" (the square used by opponent for double attack). Since in the specification syntax, we can also model responses to opponent's strategy we can make assertions of the form:

– If the opponent is trying to double attack using a knight then defend the forking square.

Pin: A pin occurs when one of the pieces is attacking the enemy king with some other enemy piece blocking its way. In this situation the blocking piece cannot move since its movement will expose the king. The most important consequence of this is that the pinned piece can be attacked. One way to protect the pinned piece is to move other pieces in to defend it. The existence of a pin is again a local property which can be checked by examining the board. A strategy which tries to save a pinned piece can be specified as follows.

– If a piece has been pinned by the opponent then defend the pinned piece.

Admittedly, the heuristics considered here are quite simplified. This is just to give a brief overview of the kind of reasoning that is possible. Usually the tactics used involve multiple moves; however it is not difficult to see that this merely increases the technical details involved in the specification.

7 Discussion

An important aspect which has not been touched upon in this account is the notion of coalitions, and looking at strategic properties by which a coalition can bring about a certain outcome. Reasoning about coalitions in game logic has been studied extensively by Marc Pauly [Pau01].

The work we have presented here on structured strategies can (in a broad sense) be compared to the notion of imperfect information. In classical game theory, since players are assumed to be rational, imperfect information arises due lack of knowledge regarding opponents' moves. Here we are dealing with a situation where the lack of knowledge is with respect to strategies rather than the moves made in the history of the play. This line of work is particularly applicable when we do not make the assumption of common knowledge of rationality (CKR). CKR assumes that a player knows all strategies that are available to him as well as his opponents. As mentioned in the previous sections, it could very well be the case that the the game is too large to explicitly analyse

all strategies. In this situation, players will tend to act irrationally. However, if we know that the opponents are of a certain "type" and the kind of strategies they employ, then this information can be explicitly used in the strategic response of the player.

Another interesting line of work would be to bring in the notion of expectation into formal strategic reasoning as, for instance, advocated by [AD05]. This is lacking in Game logic as well. In the current approach the only information that a player has about his opponent is what he has seen in the past history of play, which is unsatisfactory. Players typically begin with some expectation of the kind of strategy that the opponent plays and revise this expectation depending on what they see in the history of the play. Such problems are extensively studied in classical game theory literature where Bayesian revision of priors is a standard technique. Logical analysis of a similar kind, but using more abstract notions like "likelihood" ([HR87]), need to be explored in depth.

References

[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.

[All98] V. Allis. A knowledge based approach of connect four. Master's thesis, Vrije Universiteit, October 1998.

[Ber03] Dietmar Berwanger. Game logic is strong enough for parity games. *Studia logica*, 75(2):205–219, 2003.

[Ber07] Dietmar Berwanger. Admissibility in infinite games. In *STACS 2007 – Symposium on Theoretical Aspects of Computer Science, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the Americal Mathematical Society*, 138:295–311, 1969.

[CJM04] K. Chatterjee, M. Jurdzinski, and R. Majumdar. On nash equilibria in stochastic games. In *Proceedings of the 13th Annual Conference of the European Association for Computer Science Logic, CSL 2004*, volume 3210 of *LNCS*, pages 26–40. Springer-Verlag, 2004.

[CR] Crafty. `ftp://ftp.cis.uab.edu/pub/hyatt`.

[EJ91] A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, FOCS'91*, pages 368–377. IEEE Computer Society Press, 1991.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In *Proceedings of the 5th International Conference on Computer Aided Verification, CAV'93*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer-Verlag, 1993.

[Gal79] David Gale. The game of hex and brouwer fixed-point theorem. *The American Mathematical Monthly*, 86:818–827, 1979.

[GC] GNU Chess. `http://www.gnu.org/software/chess/`.

[GH82] Y. Gurevich and L. Harrington. Trees, automata and games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC'82*, pages 60–65. ACM Press, 1982.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, October 2002.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. The MIT Press, October 2000.

[HR87] Joseph Y. Halpern and Michael 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.

[HvdHMW03] Paul Harrenstein, Wiebe van der Hoek, John-Jules Meyer, and Cees Witteven. A modal characterization of nash equilibrium. *Fundamenta Informaticae*, 57:2–4:281–321, 2003.

[Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Kuh53] H.W. Kuhn. Extensive games and the problem of information. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games*, volume 2, pages 193–216. Princeton University Press, 1953.

[Lev90] D. N. L. Levy. *How Computers Play Chess*. W H Freeman & Co., 1990.

[LJ64] C.E. Lemke and J.T. Howson Jr. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics*, 12:413–423, 1964.

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

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149–184, 1993.

[Nas50] J.F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:89–93, 1950.

[Par85] Rohit Parikh. The logic of games and its applications. *Annals of Discrete Mathematics*, 24:111–140, 1985.

[Pau00] Marc Pauly. Game logic for game theorists. Technical report, CWI, 2000.

[Pau01] Marc Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, October 2001.

[Rab69] Micheal O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[RN02] S. J. Russell and P .Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 2002.

[RS06] R. Ramanujam and Sunil Simon. Axioms for composite strategies. *Proceedings of Logic and Foundations of Games and Decision Theory*, July 2006.

[RS07] R. Ramanujam and Sunil Simon. Structured strategies in games on graphs. In Eric Grädel, Jörg Flum, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives.*, volume 2 of *Texts in Logic and Games*, pages 567–587. Amsterdam University Press, 2007.

[SBB+07] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. In *Science*, volume 317, pages 1518–1522. American Association for the Advancement of Science, September 2007.

[Str93] Philip D. Straffin. *Game Theory and Strategy*. The Mathematical Association of America, 1993.

[Umm06] M. Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In *n Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *LNCS*, pages 212–223. Springer-Verlag, 2006.

[vB01] Johan van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[vdHJW05] Wiebe van der Hoek, Wojtek Jamroga, and Michael Wooldridge. A logic for strategic reasoning. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 05)*, pages 157–164, 2005.

[vNM47] J. von Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, 1947.

[vS02] B. von Stengel. Computing equilibria for two-person games. *Handbook of Game Theory*, 3:1723–1759, 2002.

[Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Soc. Math. Belg.*, 8(2), May 2001.

[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.

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