Structured strategies in games on graphs

R. Ramanujam and Sunil Simon

[1] The Institute of Mathematical Sciences
C.I.T. Campus, Chennai 600 113, India.
{jam,sunils}@imsc.res.in

1 Summary

We discuss strategies in non-zero sum games of perfect information on graphs. The study of non-zero sum games on graphs is motivated by the advent of computational tasks on the world-wide web and related security requirements which have thrown up many interesting areas of interaction between game theory and computer science. For example, signing contracts on the web requires interaction between principals who do not know each other and typically distrust each other. Protocols of this kind which involve *selfish agents* can be easily viewed as strategic games of imperfect information. These are complex interactive processes which critically involve players reasoning about each others' strategies to decide on how to act. In the case of interactng web services, these games involve infinite plays as well. Developing a game theoretic computational study of such interactions is an interesting challenge. Admittedly, these are games of partial information, but a theoretical analysis is interesting even in the more restricted case of perfect information.

On one hand, zero sum games on graphs have been extensively studied in logic and automata theory ([GTW02]), and on the other, a rich theory of non-zero sum *matrix form* games has been developed by game theorists ([OR94]). We call graph games large, to indicate that plays consist of (long) sequences of moves, whereas matrix form games are termed small, in the sense that a play is typically one simultaneous move. We can have matrix form presentations for sequential plays as well, but not very usefully for analysis.

While one talks of winning strategies in win / loss games, when players have overlapping objectives, we consider the best response each player can offer to moves of other players. In a small game which consists of both players deciding on a move simultaneously, it is best analyzed by considering pairs of moves. When we have a pair

(a, b) such that a is player 1's best response to player 2 deciding on b, as well as the other way about, they constitute a Nash equilibrium: there is no incentive for rational players to unilaterally deviate from such a decision. Thus equilibrium concepts predict rational play, and games are so designed that equilibrium behaviour achieves desired outcomes. Nash's theorem asserts the existence of equilibria in the space of randomized strategies and game theory offers similar theorems for related notions of equilibria.

Equating equilibria with rational play rests on the following analysis: at a game position a rational player would choose the best response to the opponent's strategy which (by assumption of rationality of the opponent) must be his best possible choice of move. Thus, the reasoning critically involves players reasoning about other players' strategies. When strategies consist of picking one move out of a set of possible moves, such as in small games, this is clear. When strategies use the current history of play to make a local move when the eventual outcome is not as yet determined, the situation is much less clear.

A strategy is a function from the set of partial plays to moves: it advises a player at a game position on the choice she can make. In a large game, this amounts to a complete specification of behaviour in all possible game situations. But then in such a game, one player's knowledge of the strategies employed by the other is necessarily partial. Rational play requires much finer analysis since strategies have structure that depends on the player's observations of game positions, history of play and the opponent's apparent strategies.

Such study of structure in strategies is relevant even in finite, determined, but large, zero-sum games. A classic example of such a game is the game of chess. Zermello showed in [Zer13] that chess is determined, i.e. from every game position, either there exists a (pure) strategy for one of the two players (white or black) guaranteeing that she will win or each one of the two players has a strategy guaranteeing at least a draw. However, given any game position, we do not know which of the three alternatives is the correct one. For games like Hex, it is known that the first player can force a win [Gal79] but nonetheless a winning strategy is not known. Again, in such situations, rather than be content with reasoning *about games* using the functional notion of strategies, one needs to reason *about strategies* themselves. For instance, most of the chess playing programs use heuristics which 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.

Thus we are led to the idea of strategies specified in a syntax, and composed structurally, with a player's strategies built up using assumptions about another. The notion of strategy composition is inspired by an analogous notion of *game composition* proposed by Rohit Parkh ([Par85]) who initiated the study of game structure using algebraic properties.

In this paper, we suggest that standard automata theoretic techniques can be employed to usefully specify and analyze partial strategies in non-zero games on graphs. We propose a syntactic framework for strategies in which best response can be algorithmically determined, and a simple modal logic in which we can reason about such strategies. This proposal is intended more as an illustration of such analysis; ideally, we need a "programming language" for strategies, whose structure should be determined empirically by how well they describe interesting heuristics employed in many classes of games that arise in applications mentioned above.

1.1 Related work

Automata theoretic analyses of two-player zero-sum infinite games of perfect information ([GTW02]) have led to interesting applications in the design and verification of reactive systems and in control synthesis. We use this technical machinery, but in the non-zero sum context.

As remarked earlier, the logical structure we study is inspired by propositional game logic ([Par85]). Pauly ([Pau01]) has built on this to provide interesting relationships between programs and games, and to describe coalitions to achieve desired goals. Bonnano ([Bon91]) suggested obtaining game theoretic solution concepts as characteristic formulas in modal logic. van Benthem ([vB01]) uses dynamic logic to describe games as well as (atomic) strategies.

On the other hand, the work on Alternating Temporal Logic ([AHK98]) considers selective quantification over paths that are possible outcomes of games in which players and an environment alternate moves. Here, we talk of the existence of a strategy for a coalition of players to force an outcome. [Gor01] draws parallels between these two lines of work, that of Pauly's coalition logics and alternating temporal logic. In the work of [HvdHMW03] and [vdHJW05], van der Hoek and co-authors develop logics for strategic reasoning and equilibrium concepts.

The underlying reasoning, whether explicitly described (as in game

logics) or implicit (as in automata theoretic studies) is carried out in a logic of games and the reasoning is about *existence* of strategies, rather than about strategies themselves. For instance, the existence of an appropriate strategy in sub-games is used to argue the existence of one in the given game. Moreover, most of the techniques involve win / lose games. Thus our departure consists in considering non-zero sum games and (hence) structured partial strategies.

In ([RS06]), we presented an axiomatization of the logic we discuss here. In this paper, the emphasis is more on showing how standard automata theoretic techniques can be employed to solve the associated algorithmic questions.

2 Games and strategies

We begin with a description of the game arena. We use the graphical model for extensive form turn-based games, where at most one player gets to move at each game position.

Game Arena

Let $N = \{1, 2\}$ be the set of players and $\Sigma = \{a_1, a_2, \ldots, a_m\}$ be a finite set of action symbols, which represent moves of players.

A **game arena** is a finite graph $\mathcal{G} = (W^1, W^2, \longrightarrow, w_0)$ where W^i is the set of *game positons* of player i for $i \in \{1, 2\}$. Let $W = W^1 \cup W^2$. The transition function $\longrightarrow$: $(W \times \Sigma) \rightarrow W$ is a partial function also called the move function and w_0 is the initial node of the game. Let $\bar{i} = 2$ when $i = 1$ and $\bar{i} = 1$ when $i = 2$.

Let the set of successors of $w \in W$ be defined as $\vec{w} = \{w' \in W \mid w \xrightarrow{a} w' \text{ for some } a \in \Sigma\}$. We assume that for all game positions w, $\vec{w} \neq \varnothing$.

In an arena, the play of a game can be viewed as placing a token on w_0. If player i owns the game position w_0 (i.e $w_0 \in W^i$), then she picks an action 'a' which is enabled for her at w_0 and moves the token to w' where $w_0 \xrightarrow{a} w'$. The game then continues from w'. Formally, a play in $\mathcal{G}$ is an infinite path $\rho : w_0 a_0 w_1 a_1 \cdots$ where $\forall j : w_j \xrightarrow{a_j} w_{j+1}$. Let *Plays* denote the set of all plays in the arena.

Games and Winning Conditions

Let $\mathcal{G}$ be an arena as defined above. The arena merely defines the rules about how the game progresses and terminates. More interesting are the winning conditions of the players, which specify the game outcomes. Since we consider non-zero sum games, players' objectives need not be strictly conflicting, and each player has a preference re-

lation 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. Let $\preceq^i \subseteq (\mathit{Plays} \times \mathit{Plays})$ be a complete, reflexive, transitive binary relation denoting the preference relation of player i for $i \in \{1, 2\}$. Then the game G is given as, $G = (\mathcal{G}, \{\preceq^i\}_{i\in\{1,2\}})$.

In general, the preference relation need not have a finite presentation, and we restrict our attention to *finite state* preferences. (This is because in the applications we have in mind, as in network games, desired or preferred plays are easily expressed as formulas of temporal logics.) Thus, the preferences of players are presented as finite state evaluation automata, with Muller acceptance conditions.

Let $\mathcal{M} = (R, \Delta, r_0)$ be a deterministic automaton with finite set of states R, initial state $r_0 \in R$ and transition function $\Delta : R \times W \times \Sigma \to R$. The evaluation automaton is given by: $\mathcal{E} = (\mathcal{M}, \{\lhd^i\}_{i\in\{1,2\}})$ where $\lhd^i \subseteq (\mathcal{F} \times \mathcal{F})$ is a total order over $\mathcal{F} = 2^R \setminus \varnothing$ for $i \in \{1, 2\}$.

A run of $\mathcal{E}$ on a play $\rho : s_0 a_0 \cdots \in \mathit{Plays}$ is a sequence of states $\varphi : r_0 r_1 \cdots$ such that $\forall i : 0 \leq i < n$, we have $r_{i+1} = \Delta(r_i, s_i, a_i)$. Let $\mathit{inf}(\varphi)$ denote the set of states occurring infinitely often in φ. The evaluation automaton $\mathcal{E}$ induces a preference ordering on *Plays* in the following manner. Let $\rho : s_0 a_0 s_1 \cdots$ and $\rho' : s_0 a'_0 s'_1 \cdots$ be two plays. Let the run of $\mathcal{E}$ on ρ and ρ' be $\varphi : r_0 r_1 \cdots r_n$ and $\varphi' : r_0 r'_1 \cdots r'_n$ respectively. For $i \in \{1, 2\}$, we have $\rho \preceq^i \rho'$ iff $\mathit{inf}(\varphi) \lhd^i \mathit{inf}(\varphi')$. A game is presented as $G = (\mathcal{G}, \mathcal{E})$.

We will also be interested in binary evaluation automata which specify least outcomes for player i. Such a automaton is given by $\mathcal{E}^i_F$, where $F \in 2^R$: for every $F' \in 2^R$, 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.

Thus we have game arenas, with players' preference on plays. We now discuss strategies of players.

Strategies

Let $\mathcal{G}_T$ denote the tree unfolding of the arena $\mathcal{G}$. We use s, s' to denote the nodes in $\mathcal{G}_T$. 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 $i = 1, 2$. We will use μ to denote a strategy of player 1 and τ a strategy of player 2. A strategy profile $\langle \mu, \tau \rangle$ defines a unique path ρ^τ_μ in the arena $\mathcal{G}$.

In games with overlapping objectives, the common solution concept employed is that of an equilibrium strategy profile [Nas50]. A profile of strategies, one for each player, is said to be in equilibrium if no player gains by unilaterally deviating from his strategy. The notion of equilibrium can be formally defined as follows. Let μ denote a strategy of player 1 and τ denote a strategy of player 2.

- μ is the best response for τ iff $\forall \mu' \in \Omega^1$, $\rho^\tau_{\mu'} \preceq^1 \rho^\tau_\mu$.
- τ is the best response for μ iff $\forall \tau' \in \Omega_2$, $\rho^{\tau'}_\mu \preceq^2 \rho^\tau_\mu$.
- $\langle \mu, \tau \rangle$ is a Nash equilibrium iff μ is the best response for τ and τ is the best response for μ.

The natural 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 have evaluation automata presenting preferences; we now proceed to a syntax for strategies.

3 Strategy specification

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 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:

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

Syntax

Let $P^i = \{p^i_0, p^i_1, \ldots\}$ be a countable set of observables for $i \in \{1, 2\}$ and let $P = P^1 \cup P^2$. The syntax of strategy specifications is then given by:

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

where $\pi \in Strat^{\bar{i}}(P^1 \cap P^2)$ 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 $\xi, k \models \psi_2$.
- $\xi, k \models \Diamond\psi$ 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 $V : W \to 2^P$. We assume the presence of two special propositions τ_i for each $i \in \{1, 2\}$ which specify at a game position, which player's turn it is to move, i.e. $\tau_i \in V(w)$ iff w is a player i game position. 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 player i node 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 $\rho_s, s_j \models_i \sigma_2$.
- $\rho_s, s_j \models_i \pi \Rightarrow \sigma_1$ iff for all player $\bar{i}$ nodes $s_k \in \rho_s$ such that $k \leq j$, if $\rho_s, s_k \models_{\bar{i}} \pi$ then $\rho_s, s_j \models_i \sigma_1$.

Above, $\pi \in Strat^{\bar{i}}(P^1 \cap P^2)$, $\psi \in Past(P^i)$, 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.

Remarks

Note that we do not have negation in specifications. One reason is that they 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).

When we consider repeated or multi-stage games, we have strategy *switching*, whereby players receive payoffs at specified points, and depending on the outcomes, decide on what new strategies to adopt later. Then it makes sense to include specifications whereby a player conforms to a strategy until some observable change, and then switches to another strategy. In this context, we have (a form of) sequential composition as well as iteration. However, operators are best added after a systematic study of their algebraic properties. We stick to a simple presentation here since our main aim is only to describe the framework. As we will see below, any set of specifications that allows effective automaton consruction will do.

Clearly, each strategy specification defines a set of strategies. We now show that it is a *regular* set, recognizable by a finite state device. In the spirit of prescriptive game theory, we call them advice automata.

Advice Automata

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. In such a framework we can ask, given a bounded memory strategy for player 2 represented by a *deterministic* strategy automaton B, can we compute the best response for player 1?

Proposition 3.1. Given a game $G = (\mathcal{G}, \mathcal{E})$ and a deterministic advice automaton B for player 2, the best response for player 1 can be effectively computed.

The proposition is proved easily. For each $F \in 2^R$, we can construct a nondeterministic automaton A_F which explores paths of $\mathcal{G}$ as follows. It consults B to pick player 2's moves and simply guesses 1's moves. It runs the binary evaluation automaton $\mathcal{E}^1_F$ for player 1 in parallel and checks if the run is winning for player 1. Now, we can enumerate the $F \in 2^R$ in such a way that those higher in $\lhd^1$ appear earlier in the enumeration. We try automata A_F in this order.

Therefore, given an strategy profile presented as advice automaton for each of the players, we can also check if a strategy profile constitutes a Nash equilibrium. However, we are interested in strategy specifications which are partial and hence constitute nondeterministic advice automata. The following lemma relates structured strategy specifications to advice automata.

Lemma 3.2. Given a player $i \in \{1, 2\}$ and a strategy specification σ, we can construct an advice automaton $\mathcal{A}_\sigma$ such that $\mu \in \mathit{Lang}(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.

Proof. The construction of automata is inductive, on the structure of specifications. Note that the strategy is implemented principally by the output function of the advice automaton.

For a strategy specification σ, let $\mathit{SF}(\sigma)$ denote the subformula closure of σ and $\mathit{SF}_\psi(\sigma)$ denote the *Past* subformulas in σ. Call $R \subseteq \mathit{SF}_\psi(\sigma)$ an atom if it is propositionally consistent and complete:

that is, for every $\neg\gamma \in SF_\psi(\sigma)$, $\neg\gamma \in R$ iff $\gamma \notin R$, and for every $\gamma_1 \vee \gamma_2 \in SF_\psi(\sigma)$, $\gamma_1 \vee \gamma_2 \in R$ iff $\gamma_1 \in R$ or $\gamma_2 \in R$.

Let $\mathcal{AT}_\sigma$ denote the set of atoms. Let $C_0 = \{C \in \mathcal{AT}_\sigma |$ there does not exist any $\diamond\psi \in C\}$. For $C, D \in \mathcal{AT}_\sigma$,define $C \longrightarrow D$ iff for all $\diamond\psi \in SF_\psi(\sigma)$, the following conditions hold.

- $\psi \in C \Rightarrow \diamond\psi \in D$
- $\diamond\psi \in D \Rightarrow \psi \in C$ or $\diamond\psi \in C$.

We proceed by induction on the structure of σ. We construct automata for atomic strategies and compose them for complex strategies.

($\sigma \equiv [\psi \mapsto a]$): The automaton works as follows. Its states keep 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. Formally:

$\mathcal{A}_\sigma = (Q_\sigma, \delta_\sigma, o_\sigma, I_\sigma)$, where

- $Q_\sigma = \mathcal{AT}_\sigma \times \Sigma$.
- $I_\sigma = \{(C, x) | C \in C_0, V(s_0) = C \cap P_\sigma, x \in \Sigma\}$.
- For a transition $s \xrightarrow{a} s'$ in $\mathcal{G}$, we have:
  $\delta_\sigma((C, x), s, a) = \{(C', y) | C \longrightarrow C', V(s') = C' \cap P_\sigma, y \in \Sigma\}$.
- $o((C, x), s) = \begin{cases} a & \text{if } \psi \in C \\ x & \text{otherwise} \end{cases}$

We now prove the assertion in the lemma that $\mu \in \mathit{Lang}(\mathcal{A}_\sigma)$ iff $\mu \models_i \sigma$.

($\Rightarrow$) Suppose $\mu \in \mathit{Lang}(\mathcal{A}_\sigma)$. Let $T = (W^1_\mu, W^2_\mu, \longrightarrow_T, \lambda)$ be the Q-labelled tree accepted by $\mathcal{A}_\sigma$. We need to show that for all $s \in W_\mu$, we have $\rho_s, s \models \psi$ implies $\mathit{out}(s) = a$.

The following claim, easily proved by structural induction on the structure of ψ, using the definition of $\longrightarrow$ on atoms, asserts that the states of the automaton check the past requirements correctly. Below we use the notation $\psi \in (C, x)$ to mean $\psi \in C$.

Claim 3.3. For all $s \in W_\mu$, for all $\psi' \in SF_\psi(\sigma)$, $\psi' \in \lambda(s)$ iff $\rho_s, s \models \psi'$.

Assume the claim and consider any $s \in W_\mu$. From claim 3.3, we have $\rho_s, s \models \psi$ implies $\psi \in \lambda(s)$. By the definition of o, we have $o(\lambda(s), s) = a$.

($\Leftarrow$) Suppose $\mu \models_1 [\psi \mapsto a]$. From the semantics, we have $\forall s \in W^1_\mu, \rho_s, s \models \psi$ implies $out(s) = a$. We need to show that there exists a Q-labelled tree accepted by $\mathcal{A}_\sigma$. For any s let the Q-labelling be defined as follows. Fix $x_0 \in \Sigma$.

- For $s \in W^1_\mu$, let $\lambda(s) = (\{\psi' \in SF_\psi(\sigma) | \rho_s, s \models \psi'\}, out(s))$.

- For $s \in W^2_\mu$, let $\lambda(s) = (\{\psi' \in SF_\psi(\sigma) | \rho_s, s \models \psi'\}, x_0)$.

It is easy to check that $\lambda(s)$ constitutes an atom and the transition relation is respected. By the definition of o, we get that it is accepting.
($\sigma \equiv \sigma_1 \cdot \sigma_2$): 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 $\mathcal{A}_{\sigma_2}$.
($\sigma \equiv \sigma_1 + \sigma_2$): 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 we have $\mathcal{A}_\pi = (Q_\pi, \delta_\pi, o_\pi, I_\pi)$ 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 $\mathcal{A}_{\sigma'}$.

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 $s \in W^1_\mu$, we have $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_1, q_2) | q_1 \in \delta_\pi(q_\pi, s, a)$ and $q_2 \in \delta_{\sigma'}(q_{\sigma'}, s, a)\}$.

- For $s \in W^2_\mu$, we have:

  - If $o_\pi(q_\pi, s) \neq a$, then $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_{free}, a) | a \in \Sigma\}$.

- If $o_\pi(q_\pi, s) = a$, then $\delta((q_\pi, q_{\sigma'}), s, a) = \{(q_1, q_2) | q_1 \in \delta_\pi(q_\pi, s, a)$ and $q_2 \in \delta_{\sigma'}(q_{\sigma'}, s, a)\}$.

- $\delta((q_{free}, x), s, a) = \{(q_{free}, a) | a \in \Sigma\}$

The output function is defined as follows: For $s \in W^1_\mu$, $o((q_\pi, q_{\sigma'}), s) = o_{\sigma'}(q_{\sigma'}, s)$ and $o((q_{free}, x), s) = x$.

The automaton keeps simulating both $\mathcal{A}_\pi$, $\mathcal{A}_{\sigma'}$ and keeps checking if the path violates π. If so it moves into state q_{free} from which point onwards it is not constrained to follow σ'.

Q.E.D.

4 Best response

Since a strategy specification denotes a set of strategies satisfying certain propeties, notions like strategy comparison and best response with respect to strategy specifications need to be redefined.

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.

- $Better_1(\sigma, \sigma')$: if $\exists F \in 2^R, \exists \mu'$ 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.

- $Better_2(\sigma, \sigma')$: if $\exists F \in 2^R$ such that $\forall \mu'$ with $\mu' \models_i \sigma'$, $\forall \tau$ with $\tau \models_{\bar{i}} \pi$, $\rho^\tau_{\mu'}$ is winning with respect to $\mathcal{E}^i_F$ then $\forall \mu$ with $\mu \models_i \sigma$, $\forall \tau$ with $\tau \models_{\bar{i}} \pi$, 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 way of comparison.

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 $\bar{i}$ and $\mathcal{E}^i_F$ a binary evaluation automaton for player i.

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

- Is it the case that for all strategies of player i conforming to σ, as long as player $\bar{i}$ is playing a strategy conforming to π, the result will be a valid play which is winning for i with respect to $\mathcal{E}_F^i$ (abbreviated as $\forall\sigma, \forall\pi : \mathcal{E}_F^i$)?

We call this the *verification* question. The *synthesis* question is given π and $\mathcal{E}_F^i$ to construct a specification σ such that $\exists\sigma, \forall\pi : \mathcal{E}_F^i$ 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 player $\bar{i}$ and $F \in \mathcal{F}$, synthesize (if possible) a specification σ for i such that $\exists\sigma, \forall\pi : \mathcal{E}_F^i$ holds.

- Given a strategy specification π for $\bar{i}$, synthesize a specification σ such that σ is the best response to π.

The main theorem of the paper is the following assertion.

Theorem 4.1. Given a game $G = (\mathcal{G}, \mathcal{E})$ and a strategy specification π for player $\bar{i}$,

1. The verification problem of checking whether for a player i strategy specification σ and a binary evaluation automaton $\mathcal{E}_F^i$, if $\exists\sigma, \forall\pi : \mathcal{E}_F^i$ and $\forall\sigma, \forall\pi : \mathcal{E}_F^i$ 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.

3. For a specification σ, checking if σ is the best response to π is decidable.

4. It is possible to synthesize a deterministic advice automaton $\mathcal{A}_i$ such that $\mathcal{A}_i$ is the best response to π.

Proof. Without loss of generality we assume $i = 1$, $\bar{i} = 2$ and σ, π to be the strategy specification for player 1 and 2 respectively.

For an advice automaton $\mathcal{A}_i = (Q_i, \delta_i, I_i, o_i)$, we define the restriction of $\mathcal{G}$ with respect to $\mathcal{A}_i$ to be $\mathcal{G} \upharpoonright A_i = (U, \longrightarrow_i, S_i)$ where $U = W \times Q_i$ and $S_i = \{s_0\} \times I_i$. In U, the nodes are partitioned in the obvious way. i.e. $u = (s, q) \in U^i$ iff $s \in W^i$. The transition relation $\longrightarrow_i : U \times \Sigma \to U$ is defined as,

- $(s, q) \xrightarrow{a}_i (s', q')$ iff $s \xrightarrow{a} s'$, $q' \in \delta_i(q, s, a)$ and ($s \in W^i$ implies $o_i(q, s) = a$).

For a node $u = (s, q) \in U$, let $enabled(u) = \{a | \exists (s', q') \in U$ with $(s, q) \xrightarrow{a} (s', q')\}$. Note that for all $u \in U^i$, $|enabled(u)| = 1$

$\mathcal{G} \upharpoonright \mathcal{A}_\pi$ is the arena restricted with π. i.e. all strategies of player 2 in $\mathcal{G} \upharpoonright \mathcal{A}_\pi$ conform to π. The game arena $\mathcal{G} \upharpoonright \mathcal{A}_\pi$ is no longer deterministic. However, for any player 2 node in $\mathcal{G} \upharpoonright \mathcal{A}_\pi$ there is exactly one action enabled (i.e. $\left|\{a \in \Sigma \mid \exists\, u' \text{ with } u \xrightarrow{a} u'\}\right| = 1$).

(1): To check if $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ holds, we build a non-deterministic tree automaton $\mathcal{T}$ which runs on $\mathcal{G} \upharpoonright \mathcal{A}_\pi$. For a 1 node, it guesses an action "a" which conforms to σ and branches out on all a edges. For a 2 node, there is only one action enabled in $\mathcal{G} \upharpoonright \mathcal{A}_\pi$, call the action b. The automaton branches out on all b labelled edges. $\mathcal{T}$ runs $\mathcal{E}^1_F$ in parallel to verify that all plays thus constructed are winning for 1 with respect to $\mathcal{E}^1_F$. If $\mathcal{T}$ has an accepting run, then $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ holds in $\mathcal{G}$. The details are as follows.

Consider $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ in $\mathcal{G}$. According to the proof of lemma 3.2, construct the advice automaton $\mathcal{A}_\sigma = (Q_\sigma, \delta_\sigma, I_\sigma, o_\sigma)$ and $\mathcal{A}_\pi = (Q_\pi, \delta_\pi, I_\pi, o_\pi)$. Let $\mathcal{E}^i_F = (\mathcal{M}, \{\lhd^i\}_{i \in \{1,2\}})$ with $\mathcal{M} = (R, \Delta, r_0)$.

Let $\mathcal{G}' = \mathcal{G} \upharpoonright \mathcal{A}_\pi = (U, \longrightarrow_\pi, S_\pi)$. Its easy to see that all player 2 strategies in $\mathcal{G}'$ is accepted by $\mathcal{A}_\pi$. Therefore we have $\exists\sigma, \forall\pi : \mathcal{E}^i_F$ holds in $\mathcal{G}$ iff there is a strategy μ accepted by $\mathcal{A}_\sigma$ such that for each strategy τ of 2 in $\mathcal{G} \upharpoonright \mathcal{A}_\pi$, the resulting path is winning for 1 with respect to $\mathcal{E}^i_F$. We give a nondeterministic top down tree automaton $\mathcal{T}$, which checks this property. Since S_π in general has more than one element, we add a new position called *root* and for all $u \in S_\pi$ add edges labelled with ϵ between *root* and u.

Formally, the tree automaton $\mathcal{T} = (Q, \delta, I)$ where $Q = (Q_\sigma \times R) \cup \{q_{root}\}$ and $I = q_{root}$. For $\mathcal{T}$ in a state q, reading node u, $\delta(q, u) = \langle (q_1, a, 1), (q_2, a, 2) \rangle$ means the automaton will branch out

into two copies, on the first a successor it goes into state q_1 and the second it goes into state q_2. For a node $u = (s, q_\pi)$, let $\overrightarrow{u}$ $\lceil a$ have k elements and let the successors be ordered in some way. The transition relation is defined as follows:

- If $u \in U^1$, then $\delta((q,r),u) = \{\langle((q',r'),a,1),\ldots,((q',r'),a,k)\rangle \mid o_\sigma(q,s) = a, q' \in \delta_\sigma(q,s,a)$ and $r' = \Delta(r,s,a)\}$.
- If $u \in U^2$, then $\delta((q,r'),u) = \{\langle((q',r'),a,1),\ldots,((q',r'),a,k)\rangle \mid q' \in \delta_\sigma(q,s,a)$ and $r' = \Delta(r,s,a)\}$.
- If $u = root$, then $\delta(q_{root},u) = \{\langle((q_0,r_0),\epsilon,1),\ldots,((q_0,r_0),\epsilon,k)\rangle \mid q_0 \in I_\sigma\}$.

To check if $\forall\sigma,\forall\pi : \mathcal{E}^i_F$ holds, it suffices to check if all plays in $(\mathcal{G} \upharpoonright \mathcal{A}_\pi) \upharpoonright \mathcal{A}_\sigma$ is winning for 1 with respect to $\mathcal{E}^1_F$. This can be done easily.

(2): We want a deterministic advice automaton $\mathcal{A}_1$ which ensures that for all strategies of 2 conforming to π the play is "winning" for player 1. We construct a tree automaton $\mathcal{T}$ which mimics the subset construction to synthesize $\mathcal{A}_1$. The states of $\mathcal{T}$ are the subsets of states of $\mathcal{A}_\pi$. At game position of player 1, it guesses a move and for every player 2 game position, it branches out on all the action choices of $\mathcal{A}_\pi$ where for each move the resulting new state is the subset of states given by the nondeterministic transition relation of $\mathcal{A}_\pi$. $\mathcal{T}$ runs $\mathcal{E}^1_F$ in parallel and checks if all paths constitutes a valid play and that the play is winning for 1 with respect to $\mathcal{E}^1_F$. If there is an accepting run for $\mathcal{T}$, then constructing $\mathcal{A}_1$ is easy. The state space of $\mathcal{A}_1$ is the set of all subsets of the states of $\mathcal{A}_\pi$. The transition relation is derived from the usual subset construction performed by $\mathcal{T}$. The output function basically follows the accepting run of $\mathcal{T}$.

Let $\mathcal{A}_\pi = (Q_\pi, \delta_\pi, I_\pi, o_\pi)$ be the advice automaton corresponding to the strategy specification π. Let $\mathcal{B} = (Q_b, \delta_b, I_b, G)$. We extend the transition relation δ_π as follows. For a set $X \subseteq Q_\pi$, $\delta_\pi(X,s,a) = \cup_{q\in X}\delta_\pi(q,s,a)$. Let $\mathcal{T} = (Q,\delta,q_0)$ be the tree automaton where $Q = 2^{Q_\pi} \times R$ and the initial state $q_0 = I_\pi \times \{r_0\}$ is the set of all initial states of $\mathcal{A}_\pi$. For a tree automaton in state q reading node s of the tree, $\delta(q,s) = \langle(q_1,a),(q_2,b)\rangle$ means that the automaton will branch out into two copies , on the a labelled outgoing edge of s it goes into state q_1 and on the b labelled outgoing edge, it goes into state q_2.

For game position s, and an automaton state $q = (\{q^1_\pi,\ldots,q^k_\pi\},r)$, the transition relation is defined as follows:

- if $s \in W^1$: $\delta(q,s) = \{\langle((p,r'),a)\rangle | \exists s \xrightarrow{a} s' \text{ in } \mathcal{G}, p = \delta_\pi(q,s,a)$ and $r' = \Delta(r,s,a)\}$.

- if $s \in W^2$: Let $\{a_1,\ldots,a_k\} = \{o_\pi(q_\pi^1),\ldots,o_\pi(q_\pi^k)\}$. $\delta(q,s) = \{\langle((p_1,r_1),a_1),\ldots,((p_k,r_k),a_k)\rangle | p_i = \delta_\pi(q,s,a_i)$ and $r_i = \Delta(r,s,a_i)\}$.

If $\mathcal{T}$ has a successful run on $\mathcal{G}$, then let T_π be the run tree with λ being the labelling function from game positions to Q. We build the advice automaton for 1 from this tree. The advice automaton $\mathcal{A}_1 = (q_1, \delta_1, q_1^0, o_1)$ where $Q_1 = 2_\pi^Q$, $q_1^0 = I_\pi$, $\delta_1(q_1,s,a) = q'$ if in T_π we have $s \xrightarrow{a} s'$ where $\lambda(s) = (q,r)$ and $\lambda(s') = (q',r')$. By definition of the transition function of $\mathcal{T}$, δ_1 is deterministic. The output funciton o_1, for each of the 1 nodes is dictated by the guess made by $\mathcal{T}$ on the succcessful run T_π.

(3): Given σ and π to check if σ is the best response to π, we use the tree automaton construction in (1) with a slight modification.

We enumerate the elements of 2^R in such a way that those higher in $\lhd^1$ appear earlier in the enumeration. For each F, we construct a tree automaton as in (1), the only difference being that the guesses made by $\mathcal{T}$ at player 1 game positions are not restricted by σ. $\mathcal{T}$ runs $\mathcal{E}_F^1$ in parallel to check if player 1 can ensure F for all choices of 2 which conform to π. Since the evaluation automaton is "complete", the play eventually settles down in one of $F' \in 2^R$. Therefore, as we try elements of 2^R in order, the tree automaton succeeds for some $\mathcal{E}_{F'}^1$. This gives us the "best" outcome which player 1 can guarantee. We then check if $\exists\sigma, \forall\pi : \mathcal{E}_{F'}^1$ holds in $\mathcal{G}$. If it does then $\mathcal{A}_\sigma$ is a best response to $\mathcal{A}_\pi$.

This also implies that we can check whether a strategy profile (presented as advice automata) constitutes a Nash equilibrium.

(4) is similar to (3). We enumerate 2^R and find the "best" outcome that can be achieved and using the synthesis procedure, synthesize an advice automaton for this outcome. Q.E.D.

5 A strategy logic

We now discuss how we may reason about 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). 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.

Syntax

Let $P^i = \{p_0^i, p_1^i, \ldots\}$ be a countable set of proposition symbols where $\tau_i \in P_i$, for $i \in \{1, 2\}$. Let $P = P^1 \cup P^2$. τ_1 and τ_2 are intended to specify, at a game position, which player's turn it is to move. Further, the logic is parametrized by the finite alphabet set $\Sigma = \{a_1, a_2, \ldots, a_m\}$ of players' moves and we only consider game arenas over Σ.

The syntax of the logic is given by:

$$\Pi := p \in P \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \langle a\rangle\alpha \mid \diamondsuit\alpha \mid (\sigma)_i : c \mid \sigma \rightsquigarrow_i \beta$$

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

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.

Semantics

The models for the logic are extensive form game trees along with a valuation function. A model $M = (T, V)$ where $T = (S, \longrightarrow, s_0)$ is a game tree obtained by the unfolding of the arena $\mathcal{G}$, and $V : S \to 2^P$ is the valuation function.

Given a game tree T and a node s in it, let $\rho^s_{s_0} : s_0 \stackrel{a_1}{\Longrightarrow} s_1 \cdots \stackrel{a_m}{\Longrightarrow} s_m = s$ denote the unique path from s_0 to 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 at a node s (denote $\sigma(s)$). For a strategy specification $\sigma \in \mathit{Strat}^i(P^i)$ and a node s we define $\sigma(s)$ as follows:

- $\mathit{null}(s) = \Sigma$.
- $[\psi \mapsto a]^i(s) = \begin{cases} \{a\} & \text{if } s \in W^i \text{ and } \rho^s_{s_0}, m \models \psi \\ \Sigma & \text{otherwise.} \end{cases}$
- $(\sigma_1 + \sigma_2)(s) = \sigma_1(s) \cup \sigma_2(s)$.

- $(\sigma_1 \cdot \sigma_2)(s) = \sigma_1(s) \cap \sigma_2(s)$.
- $(\pi \Rightarrow \sigma)(s) = \begin{cases} \sigma(s) & \text{if } \forall j : 0 \leq j < m,\ a_j \in \pi(s_j) \\ \Sigma & \text{otherwise.} \end{cases}$

We say that a path $\rho_s^{s'} : s = s_1 \stackrel{a_1}{\Longrightarrow} s_2 \cdots \stackrel{a_{m-1}}{\Longrightarrow} s_m = s'$ 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.

Proposition 5.1. Given a strategy μ for player i along with a specification σ, $\mu \models_i \sigma$ (as defined in section 3) iff for all player i nodes $s \in \mu$ we have $out(s) \in \sigma(s)$.

For a game tree T, a node s let T_s denote the tree which consists of the unique path $\rho_{s_0}^s$ and the subtree rooted at s. For a strategy specification $\sigma \in Strat^i(P^i)$, we define $T_s \restriction \sigma = (S_\sigma, \Longrightarrow_\sigma, s_0)$ to be the least subtree of T_s which contains the unique path from s_0 to s and satisfies the following property.

- For every s' in S_σ such that $s \Longrightarrow_\sigma^* s'$,
  - s' is an i node: $s' \stackrel{a}{\Longrightarrow} s''$ and $a \in \sigma(s') \Leftrightarrow s' \stackrel{a}{\Longrightarrow}_\sigma s''$.
  - s' is an $\bar{i}$ node: $s' \stackrel{a}{\Longrightarrow} s'' \Leftrightarrow s' \stackrel{a}{\Longrightarrow}_\sigma s''$.

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 $\rho_{s_0}^s$ be $s_0 \stackrel{a_0}{\Longrightarrow} s_1 \cdots \stackrel{a_{m-1}}{\Longrightarrow} s_m = s$.

- $M, s \models p$ 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 $s' \in W$ such that $s \stackrel{a}{\rightarrow} s'$ and $M, s' \models \alpha$.
- $M, s \models \Diamond\!\!\!\!-\alpha$ 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 $s \Longrightarrow_\sigma^* s'$ in $T_s \restriction \sigma$, we have $M, s' \models \beta \wedge (\tau_i \supset enabled_\sigma)$.

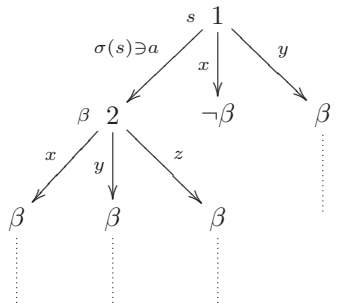

FIGURE 1.

where $enabled_\sigma \equiv \bigvee_{a \in \Sigma} (\langle a \rangle True \wedge (\sigma)_i : a)$.

Figure 1 illustrates the semantics of $\sigma \rightsquigarrow_1 \beta$. It says, for an 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 such that $M, s_0 \models \alpha$. A formula α is said to be valid iff for all models M, we have $M, s_0 \models \alpha$.

Truth Checking

The truth checking problem is given a model $M = (T, V)$ and a formula α_0, determine whether $M, s_0 \models \alpha_0$. The following theorem shows the decidability of the truth checking problem.

Theorem 5.2. Given a model $M = (T, V)$ and a formula α_0, we can construct a nondeterministic Büchi tree automaton $\mathcal{T}_{\alpha_0}$ such that $M, s_0 \models \alpha_0$ iff $\mathcal{T}_{\alpha_0}$ has an accepting run on M.

Proof. Let $\{\sigma_1, \ldots, \sigma_m\}$ be the strategy specification formulas appearing in α_0 and $\mathcal{A}_{\sigma_1}, \ldots \mathcal{A}_{\sigma_m}$ be the advice automata corresponding to the specifications. The tree automaton keeps track of the atoms (locally consistent sets of subformulas) of α_0 and the states of each of the advice automata. At any game position, it guesses a new atom which is consistent with the game position and a state for each of the advice automaton from its transition relation. For the subformula $(\sigma)_i : a$ in the atom, it only needs to check if a is the action dictated by the output function of the advice automaton for σ. 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 will be satisfied on that branch. The Büchi acceptence condition is simply all the states where the "requirement set" U is empty.

We will find some abbreviations useful:

- $inv_i^\sigma(a,\beta) = (\tau_i \wedge (\sigma)_i : a) \supset [a](\sigma \rightsquigarrow_i \beta)$ denotes the fact that after an "a" move by player i which conforms to σ, $\sigma \rightsquigarrow_i \beta$ continues to hold.

- $inv_{\bar{i}}^\sigma(\beta) = \tau_{\bar{i}} \supset [N](\sigma \rightsquigarrow_i \beta)$ says that after any move of $\bar{i}$, $\sigma \rightsquigarrow_i \beta$ continues to hold.

- $enabled_\sigma = \bigvee_{a \in \Sigma} (\langle a \rangle True \wedge (\sigma)_i : a)$.

For a formula α, let $SF(\alpha)$ denote the subformula closure of α. In addition to the usual downward closure we also require that $\sigma \rightsquigarrow_i \beta \in SF(\alpha)$ implies $enabled_i, inv_i^\sigma(a,\beta), inv_{\bar{i}}^\sigma(\beta), \beta \in SF(\alpha)$. Call $C \subseteq SF(\alpha)$ an atom if it is propositionally consistent and complete, in addition we require the following to hold.

- $\sigma \rightsquigarrow_i \beta \in C \Rightarrow enabled_\sigma, inv_i^\sigma(a,\beta), inv_{\bar{i}}^\sigma(\beta) \in C$.

- $\neg(\sigma \rightsquigarrow_i \beta) \in C \Rightarrow (\neg enabled_\sigma$ or $\neg\beta) \in C$ or $(\langle X \rangle \neg(\sigma \rightsquigarrow_i \beta)) \in C$.

Let $\mathcal{AT}_\alpha$ denote the set of atoms. Let $C_0 = \{C \in \mathcal{AT}_\alpha |$ there does not exist any $\Diamond\gamma \in C\}$. For $C, D \in \mathcal{AT}_\alpha$, define $C \xrightarrow{a} D$ iff for all $\Diamond\gamma \in SF(\alpha)$, the following conditions hold.

- $\gamma \in C \Rightarrow \Diamond\gamma \in D$.

- $\Diamond\gamma \in C \Rightarrow \gamma \in C$ or $\Diamond\gamma \in C$.

- $[a]\gamma \in C \Rightarrow \gamma \in D$.

Let $\{\sigma_1, \ldots, \sigma_m\}$ be the strategy specification formulas appearing in α_0 and let $\mathcal{A}_{\sigma_1}, \ldots \mathcal{A}_{\sigma_m}$ be the advice automata corresponding to the specifications. The tree automata $\mathcal{T} = (Q, \delta, I, F)$ where $Q \subseteq (\mathcal{AT}_{\alpha_0} \cup reject) \times (2^{SF(\alpha_0)})^3 \times Q_{\sigma_1} \times \ldots \times Q_{\sigma_m}$ such that $(C, U, Z, Y, q_1, \ldots, q_m) \in Q$ iff $(\sigma)_i : a, \tau_i \in C \Rightarrow o_\sigma(q_\sigma) = a$. 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. The set of initial states $I = \{(C, U, Z, Y, q_1^0, \ldots, q_m^0) | C \in C_0, V(s_0) = C \cap P_{\alpha_0}, U = \varnothing, Z = \varnothing$ and $q_i^0 \in I_{\sigma_i}\}$, $Y = \{\langle a\rangle\alpha | a \in \Sigma$ and $\langle a\rangle\alpha \in C\}$.

For a node s, let $s_1, \ldots, s_k$ be its successors in $\mathcal{G}$ with $s \xrightarrow{a_j} s_j$ for $1 \leq j \leq k$. For a state $q = (C, U, Z, Y, q_1, \ldots, q_m)$ at s, the automaton guesses a partition of $U = U_1 \cup \ldots \cup U_k$ and a partition $Y = Z_1 \cup \ldots \cup Z_k$. The transition relation is then defined as: $\langle((C_1, U_1', Z_1, Y_1, q_1^1, \ldots q_m^1), a_1), \ldots ((C_k, U_k', Z_1, Y_1, q_1^k, \ldots, q_m^k), a_k)\rangle \in \delta((C, U, q_1, \ldots, q_m), s)$ iff

- $C_j =$ *reject* if there exists $\langle a\rangle\alpha \in Z_j$ such that $\alpha \notin C_j$ or $a_j \neq a$
- For $1 \leq j \leq k$, $C \xrightarrow{a_j} C_j$ and $V(s_j) = C_j \cap P_{\alpha_0}$.
- For $1 \leq j \leq k$, $1 \leq r \leq m$, $q_r^j \in \delta_r(q_r, s, a_j)$.
- $U_j' = \begin{cases} \{\sigma \rightsquigarrow_i \beta \in U_j \mid \beta, \mathit{enabled}_\sigma \in C_j\} & \text{if } U \neq \varnothing \\ \{\sigma \rightsquigarrow_i \beta \in C_j \mid \beta, \mathit{enabled}_\sigma \in C_j\} & \text{if } U = \varnothing \end{cases}$
- $Y_j = \{\langle a\rangle\alpha \mid \langle a\rangle\alpha \in C_j\}$

Once the automaton reaches the *reject* state then it remains in that state for all transitions. The Büchi acceptence condition is, $F = \{q = (C, U, Z, Y, q_1, \ldots, q_m) \in Q \mid U = \varnothing$ and $C \in \mathcal{AT}_{\alpha_0}\}$. Q.E.D.

Complexity of truth checking

For the given formula α_0, let $|\alpha_0| = n$. The states of the tree automaton are the atoms of α_0 and the states of each of the advice automaton. Since the number of strategy specifications occuring in α_0 is bounded by the size of α_0, the size of the tree automaton $|\mathcal{T}| = \mathcal{O}(n \cdot 2^n)$. Let $\mathcal{T}_\mathcal{G}$ denote the tree automaton accepting $\mathcal{G}$. We want to check for emptiness of $\mathcal{T} \cap \mathcal{T}_\mathcal{G}$. Since $\mathcal{T}$ is a Büchi tree automaton this gives us a total time complexity of $\mathcal{O}(2^n)$.

References

[AHK98] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[Bon91] G. Bonanno. The logic of rational play in games of perfect information. *Economics and Philosophy*, 7:37–65, 1991.

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

[Gor01] V. Goranko. Coalition games and alternating temporal logics. *Proceedings of 8th conference on Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 259–272, 2001.

[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.

[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.

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

[OR94] M.J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994.

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

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

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

[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.