Reasoning about Natural Strategic Ability

Vadim Malvone

Université d'Evry

December 7, 2018

Preface

System Correctness

- A very important problem in critical systems:
  - Safety: errors can cost lives (e.g. Therac-25).
  - Mission: errors can cost in terms of objectives (e.g. Arianne 5).
  - Business: failure can cost in loss of money (e.g. Denver airport).
- In such systems failure is not an option.

Preface

System Correctness

- A very important problem in critical systems:
  - Safety: errors can cost lives (e.g. Therac-25).
  - Mission: errors can cost in terms of objectives (e.g. Arianne 5).
  - Business: failure can cost in loss of money (e.g. Denver airport).
- In such systems failure is not an option.

Formal verification

- It studies in depth system correctness.
- It is based on mathematic/logical methods.
- One of the most important contributions is the Model Checking.

Model checking (1)

$M \models \varphi$

There are three fundamental parts:

- M : modeling a system;
- φ : specifying a property;
- $\models$: verifying that the model M satisfies the property φ.

Model checking (2)

Model

A Kripke Structure is a tuple $K = \langle \mathrm{AP}, \mathrm{St}, S_0, R, L \rangle$, where:

- AP is a set of atomic propositions;
- St is a set of states;
- $S_0 \subseteq \mathrm{St}$ is a set of initial states;
- $R \subseteq \mathrm{St} \times \mathrm{St}$ is a transition relation;
- $L : \mathrm{St} \to 2^{\mathrm{AP}}$ is a labeling function.

Model checking (3)

Specification

- Temporal logics allow to describe the order of the events without define the time explicitly.
- The main temporal logics:
    - LTL [Pnu77](Linear Temporal Logic), in a computation the events are totally ordered.
    - CTL [CE81] (Computation Tree Logic), in a computation the events are partially ordered.

Model checking (4)

Verification

Given a Kripke structure K and a specification φ, the problem of the model checking consists to verify if $K, s \models \varphi$, for each initial state s.

Open Systems (1)

Key aspects

- There are many agents (players) interacting among them.
- Each agent has a set of *strategies*.
- A *strategy* is a conditional plan that at each step of the game prescribes an action.
- The composition of strategies, one for each player, induces an unique computation.

Open Systems (2)

Model

- A game structure is a tuple $M =< AP, St, s_I, Ac, Ag, tr >$, where:
  - AP is a set of atomic propositions;
  - St is a set of states;
  - $s_I \in S$ is a designated initial state;
  - Ac is a set of actions;
  - Ag is a set of agents;
  - tr is a transition function.
- Depending on the interaction between the agents:
  - *Turn-based* $\Rightarrow$ The states of the game are partitioned between the agents, then the owner of a state determines the move to take and thus the next state of the game.
  - *Concurrent* $\Rightarrow$ The agents choose a move (i.e, actions) simultaneously and independently, and the choices together determine the next state of the game.

Open Systems (3)

Plays and Histories

- The infinite paths starting from s_I represent all possible plays.
- The finite paths starting from s_I represent all possible histories.
- The set of all histories is denoted by H.

Open Systems (3)

Plays and Histories

- The infinite paths starting from s_I represent all possible plays.
- The finite paths starting from s_I represent all possible histories.
- The set of all histories is denoted by H.

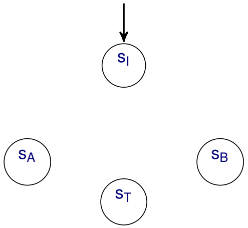
Strategies

- Depending on the memory, we distinguish between:
  - *memoryless strategies (r)* $\Rightarrow \sigma : St \to \mathrm{Ac}$;
  - *memoryfull strategies (R)* $\Rightarrow \sigma : H \to \mathrm{Ac}$.
- In r case, the players take a decision by considering the actual state of the game.
- In R case, the players take a decision by considering the history of the game.

Example: paper, rock, and scissor

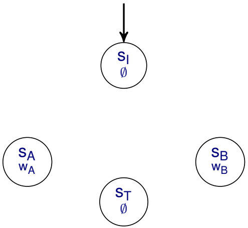
- $\mathrm{Ag} = \{\mathsf{A}\colon \mathit{Alice}, \mathsf{B}\colon \mathit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}\colon \mathit{Paper}, \mathsf{R}\colon \mathit{Rock}, \mathsf{S}\colon \mathit{Scissor}\}$

Example: paper, rock, and scissor



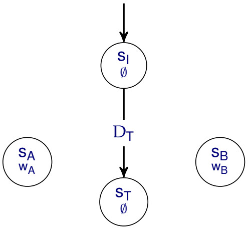
- $\mathrm{Ag} = \{\mathsf{A}\text{: } \textit{Alice}, \mathsf{B}\text{: } \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}\text{: } \textit{Paper}, \mathsf{R}\text{: } \textit{Rock}, \mathsf{S}\text{: } \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*

Example: paper, rock, and scissor



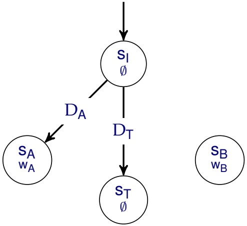
- $\mathrm{Ag} = \{\mathsf{A}\colon \textit{Alice}, \mathsf{B}\colon \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}\colon \textit{Paper}, \mathsf{R}\colon \textit{Rock}, \mathsf{S}\colon \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A\colon \textit{Alice wins}, w_B\colon \textit{Bob wins}\}$

EXAMPLE: PAPER, ROCK, AND SCISSOR



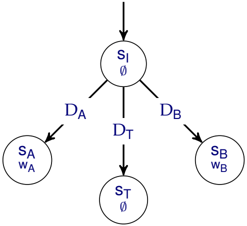
- $Ag = \{A$: *Alice*, B: *Bob*$\}$
- $Ac = \{P$: *Paper*, R: *Rock*, S: *Scissor*$\}$
- $St = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $AP = \{w_A$: *Alice wins*, w_B: *Bob wins*$\}$
- $D_T = \{(P, P), (R, R), (S, S)\}$

Example: paper, rock, and scissor



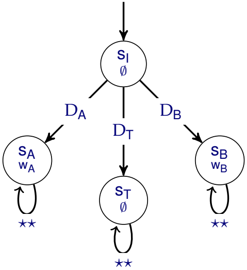
- $\mathrm{Ag} = \{\mathsf{A}: \textit{Alice}, \mathsf{B}: \textit{Bob}\}$
- $\mathrm{Ac} = \{\mathsf{P}: \textit{Paper}, \mathsf{R}: \textit{Rock}, \mathsf{S}: \textit{Scissor}\}$
- $\mathrm{St} = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $\mathrm{AP} = \{w_A: \textit{Alice wins}, w_B: \textit{Bob wins}\}$
- $D_T = \{(\mathsf{P}, \mathsf{P}), (\mathsf{R}, \mathsf{R}), (\mathsf{S}, \mathsf{S})\}$
- $D_A = \{(\mathsf{P}, \mathsf{R}), (\mathsf{R}, \mathsf{S}), (\mathsf{S}, \mathsf{P})\}$

Example: paper, rock, and scissor



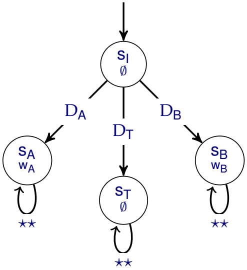
- $Ag = \{A\text{: } \textit{Alice}, B\text{: } \textit{Bob}\}$
- $Ac = \{P\text{: } \textit{Paper}, R\text{: } \textit{Rock}, S\text{: } \textit{Scissor}\}$
- $St = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $AP = \{w_A\text{: } \textit{Alice wins}, w_B\text{: } \textit{Bob wins}\}$
- $D_T = \{(P, P), (R, R), (S, S)\}$
- $D_A = \{(P, R), (R, S), (S, P)\}$
- $D_B = \{(R, P), (S, R), (P, S)\}$

Example: paper, rock, and scissor



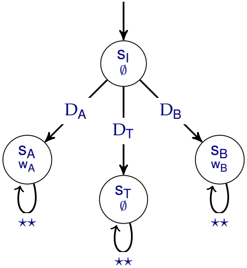
- $Ag = \{A\text{: } \textit{Alice}, B\text{: } \textit{Bob}\}$
- $Ac = \{P\text{: } \textit{Paper}, R\text{: } \textit{Rock}, S\text{: } \textit{Scissor}\}$
- $St = \{s_I, s_A, s_T, s_B\}$
- s_I: *initial state*
- $AP = \{w_A\text{: } \textit{Alice wins}, w_B\text{: } \textit{Bob wins}\}$
- $D_T = \{(P, P), (R, R), (S, S)\}$
- $D_A = \{(P, R), (R, S), (S, P)\}$
- $D_B = \{(R, P), (S, R), (P, S)\}$

Example: paper, rock and scissor



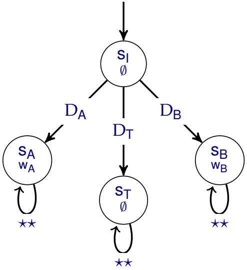
- A possible play is $\pi = s_I \cdot (s_A)^\omega$.

Example: paper, rock and scissor



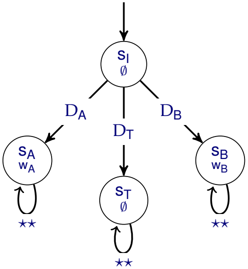
- A possible play is $\pi = s_I \cdot (s_A)^\omega$.
- An example of history is $h = s_I$.

Example: paper, rock and scissor



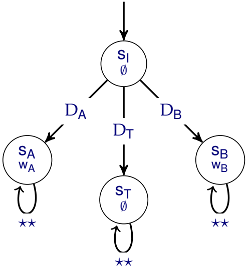
- A possible play is $\pi = s_I \cdot (s_A)^\omega$.
- An example of history is $h = s_I$.
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in H$.

Example: paper, rock and scissor



- A possible play is $\pi = s_I \cdot (s_A)^\omega$.
- An example of history is $h = s_I$.
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in H$.
- A strategy for B is $\sigma_B(h) = \mathsf{R}$, $\forall h \in H$.

EXAMPLE: PAPER, ROCK AND SCISSOR



- A possible play is $\pi = s_I \cdot (s_A)^\omega$.
- An example of history is $h = s_I$.
- A strategy for A is $\sigma_A(h) = \mathsf{P}$, $\forall h \in H$.
- A strategy for B is $\sigma_B(h) = \mathsf{R}$, $\forall h \in H$.
- σ_A and σ_B induce the play π.

Open Systems (4)

Specification

- *Internal* $\Rightarrow$ directly over the game structure (Examples: Reachability, Safety, Nash Equilibrium).
- *External* $\Rightarrow$ logics for the strategic reasoning (Examples: ATL [AHK02], Strategic Logic [MMV10]).

Open Systems Specification (1)

Alternating-Time Temporal Logic (ATL*) [AHK02]

- ATL* generalizes CTL* by replacing the path quantification with strategy quantification over coalition of agents.
- It uses the strategy modality $\langle\langle A \rangle\rangle$ and $[\![A]\!]$, with A set of agents.
- The strategies are implicit.

Open Systems Specification (1)

Alternating-Time Temporal Logic (ATL*) [AHK02]

- ATL* generalizes CTL* by replacing the path quantification with strategy quantification over coalition of agents.
- It uses the strategy modality $\langle\langle A \rangle\rangle$ and $[[A]]$, with A set of agents.
- The strategies are implicit.

Example

$\langle\langle \{\alpha, \beta\} \rangle\rangle \mathsf{G} \neg \mathit{fail}$: the agents α and β have a collective strategy such that the system does not reach the failure state, independently from the strategies of the other agents.

Open Systems Specification (2)

Model checking complexities

- ATL^* with:
  - Memoryless strategies is $PSPACE - complete$ [Sch04].
  - Memoryfull strategies is $2EXPTIME - complete$ [AHK02].
- ATL with:
  - Memoryless strategies is $PTIME - complete$ [AHK02].
  - Memoryfull strategies is $PTIME - complete$ [AHK02].

Between Mathematics and Real Life

- Strategic logics provide powerful tools to reason about multi-agent systems.
- However, strategies are *mathematical creatures*

Between Mathematics and Real Life

- Strategic logics provide powerful tools to reason about multi-agent systems.
- However, strategies are *mathematical creatures* $\Longrightarrow$ *any function* from system states to actions is fine.

Between Mathematics and Real Life

- Strategic logics provide powerful tools to reason about multi-agent systems.
- However, strategies are *mathematical creatures* $\Longrightarrow$ *any function* from system states to actions is fine.
- This makes sense for robots or programs, but not for humans!

Between Mathematics and Real Life

- Strategic logics provide powerful tools to reason about multi-agent systems.
- However, strategies are *mathematical creatures* $\Longrightarrow$ *any function* from system states to actions is fine.
- This makes sense for robots or programs, but not for humans!
- Strategies for humans should be simple in order for the person to *understand* it, *memorize* it, and *execute* it.

Natural Strategies [JMM17]

A *natural memoryless strategy* s_a for agent a is a *list of condition-action rules*

$$(cond, act)$$

such that:

- $cond$ is a boolean combination of propositions,
- act is an available action in every state $q \models cond$,
- the last pair on the list is $(\top, idle)$.

Natural Strategies: Example

Consider the following strategy for *buying a train ticket*:

1. $(\neg\mathsf{ticket} \wedge \neg\mathsf{selected},\ \mathit{select})$;
2. $(\neg\mathsf{ticket} \wedge \mathsf{selected},\ \mathit{pay})$;
3. $(\top,\ \mathit{idle})$.

NATURAL STRATEGIES: COMPLEXITY

The complexity of strategy s_a ($compl(s_a)$) can be defined by:

- Number of used propositions $\Rightarrow |dom(s_a)|$;
- Largest condition $\Rightarrow \max\{|\phi| \mid (\phi, \alpha) \in s_a\}$;
- Total size of the representation $\Rightarrow \sum_{(\phi,\alpha)\in s_a} |\phi|$.

Reasoning about Natural Ability: NatATL

Syntax

A formula in NatATL is defined as:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle^{\leq k} X\varphi \mid \langle\langle A \rangle\rangle^{\leq k} \varphi U \varphi \mid \langle\langle A \rangle\rangle^{\leq k} \varphi W \varphi.$$

where $\mathsf{p} \in AP$, $k \in \mathbb{N}$, and A is a set of agents.

Reasoning about Natural Ability: NatATL

Syntax

A formula in NatATL is defined as:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle^{\leq k} X\varphi \mid \langle\langle A \rangle\rangle^{\leq k} \varphi U \varphi \mid \langle\langle A \rangle\rangle^{\leq k} \varphi W \varphi.$$

where $\mathsf{p} \in AP$, $k \in \mathbb{N}$, and A is a set of agents.

Semantics

$M, q \models \langle\langle A \rangle\rangle^{\leq k}\gamma$ iff there is a natural strategy s_A such that $compl(s_A) \leq k$, and for each path $\lambda \in out(q, s_A)$ we have $M, \lambda \models \gamma$.

What's the Use?

Reasoning about usability, example: *ticket vending machine*

- It is not enough that a customer has a strategy to buy the ticket ($\langle\langle c\rangle\rangle F\mathsf{buy}$).
- If the strategy is too complex, people won't use it anyway.
- Instead, we should require $\langle\langle c\rangle\rangle^{\leq k} F\mathsf{buy}$ for a reasonably low k.

What's the Use?

Reasoning about usability, example: *ticket vending machine*

- It is not enough that a customer has a strategy to buy the ticket ($\langle\langle c\rangle\rangle F\mathsf{buy}$).
- If the strategy is too complex, people won't use it anyway.
- Instead, we should require $\langle\langle c\rangle\rangle^{\leq k}F\mathsf{buy}$ for a reasonably low k.

Gaming

- The designer can define the *game level* by the *complexity of the smallest winning strategy* for the player.
- Formally, the level k iff $\langle\langle a\rangle\rangle^{\leq k}F\mathsf{win} \wedge \neg\langle\langle a\rangle\rangle^{\leq k-1}F\mathsf{win}$.

Natural Strategies with Recall

- Similar to memoryless strategies, but the conditions are given by *regular expressions* over Boolean formulas.
- Example: a strategy for a *Wild West explorer*:
  1. $(\mathsf{safe}^*, \textit{digGold})$;
  2. $(\mathsf{safe}^* \cdot (\neg\mathsf{safe} \wedge \mathsf{haveGun}), \textit{shoot})$;
  3. $(\mathsf{safe}^* \cdot (\neg\mathsf{safe} \wedge \neg\mathsf{haveGun}), \textit{run})$;
  4. $(\top^* \cdot (\neg\mathsf{safe}) \cdot (\neg\mathsf{safe}), \textit{hide})$;
  5. $(\top^*, \textit{idle})$.

Relationships between Types of Natural Strategies (1)

Theorem

The following results hold in NatATL:

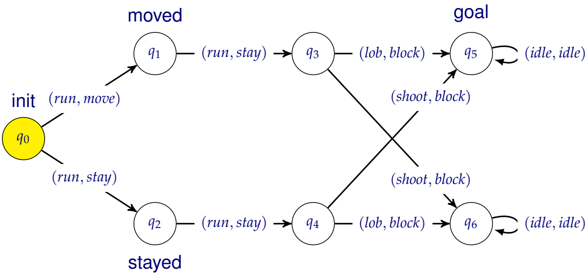
1. *For all M, q, and all formulas $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, it holds that:*

$$M, q \models_r \varphi \text{ implies } M, q \models_R \varphi$$

2. *There exist M, q, and a formula $\varphi = \langle\langle A \rangle\rangle^{\leq k}\gamma$, such that:*

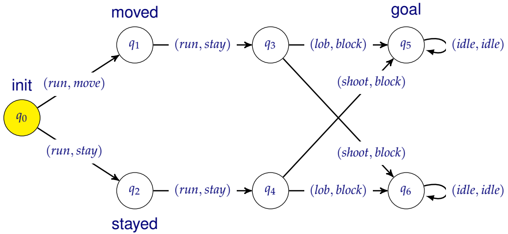
$$M, q \models_R \varphi \text{ and not } M, q \models_r \varphi$$

Example: Soccer scenario (1)



- Robot 1 is running towards the goal with the ball.
- The goalkeeper (robot 2) can either stay close to the goal line or move towards the attacker.
- Then, after one more step, the attacker can either shoot straight or lob the ball over the goalkeeper.

Example: Soccer scenario (2)

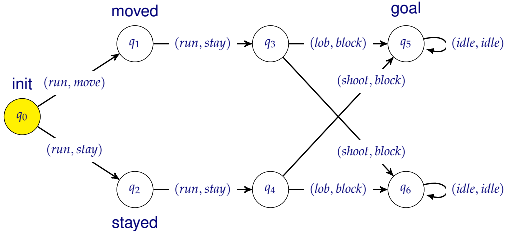


A strategy with recall for the attacker to score the goal can be:

1. (init, run);
2. (init $\cdot$ (moved $\vee$ stayed), run);
3. ($\top^* \cdot$ moved $\cdot \top$, lob);
4. ($\top^* \cdot$ stayed $\cdot \top$, $shoot$);
5. ($\top^*$, $idle$).

The complexity of the strategy is 22.

Example: Soccer scenario (3)



- Then, $\varphi = \langle\langle 1 \rangle\rangle^{\leq 22} F\mathsf{goal}$ is true for strategies with recall.
- On the other hand, φ is false for memoryless strategies.
- In fact, the formula is false for any bound k.
- To see that, recall that conditions in natural memoryless strategies can only refer to boolean properties of the current state.
- Then, it is impossible to define two different behaviors in states q_3 and q_4 within a natural memoryless strategy.

VERIFICATION OF NATURAL STRATEGIES

Model checking $NatATL_r$

- $\mathbf{P}$ for fixed or bounded k;
- $\mathbf{P}^{\mathbf{NP}} = \Delta_2^{\mathbf{P}}$-complete when k is a parameter of the problem.

Verification of Natural Strategies

Model checking $NatATL_r$

- $\mathbf{P}$ for fixed or bounded k;
- $\mathbf{P^{NP}} = \Delta_2^{\mathbf{P}}$-complete when k is a parameter of the problem.

Model checking $NatATL_R$

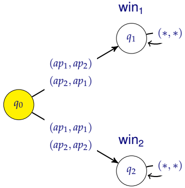
- $\Delta_2^{\mathbf{P}}$ for fixed or bounded k;
- $\mathbf{P^{NP^{NP}}} = \Delta_3^{\mathbf{P}}$ when k is a parameter of the problem.

Concurrent game with objectives [JMM]

A *concurrent game* is a tuple $G = (M, q_0, \Phi)$, where:

- M is a concurrent game structure,
- $q_0 \in St$ is a state in M,
- $\Phi : Ag \to \mathcal{L}_{LTL}$ assigns each agent with an LTL formula.

Example: Simple market scenario



Establ. Comp. \ New Comp.	ap_1	ap_2
ap_1	0, 1	1, 0
ap_2	1, 0	0, 1

- An established company (EC) and a new company (NC) have to choose the appearance for a product.
- Each company can choose between two different appearances for the product (ap_1 and ap_2).
- EC prefers the two products to look different.
- NC is better off when the products look alike.
- $\Phi_{EC} = F\mathsf{win}_1$ and $\Phi_{NC} = F\mathsf{win}_2$.

Decision problems: Surely Winning (1)

Definition

Given a concurrent game G, a subset of agents $A \subseteq Ag$, a natural number $k \in \mathbb{N}$, and a natural collective strategy s_A of A, we say that:

$$s_A \text{ is } \textit{surely winning} \text{ in } G \Leftrightarrow \forall \lambda \in out(q_0, s_A) \text{ and } a \in A\text{: } \lambda \models \Phi_a$$

Moreover, coalition A *surely wins* in G under bound k iff it has a sure winning strategy of size at most k.

Decision problems: Surely Winning (2)

Algorithm $SureWin(G, A, k)$:

$s_A = GuessStrat(G, A, k)$;
Prune M according to s_A, obtaining model M';
return $mCheck_{CTL^*}(M', q_0, \mathbf{A} \bigwedge_{i \in A} \Phi_i)$;

Decision problems: Surely Winning (2)

Algorithm $SureWin(G, A, k)$:

$s_A = GuessStrat(G, A, k)$;
Prune M according to s_A, obtaining model M';
return $mCheck_{CTL^*}(M', q_0, \mathbf{A} \bigwedge_{i \in A} \Phi_i)$;

Hint for lower bound

We show a reduction from model checking LTL.

Decision problems: Surely Winning (2)

Algorithm $SureWin(G, A, k)$:

$s_A = GuessStrat(G, A, k)$;
Prune M according to s_A, obtaining model M';
return $mCheck_{CTL^*}(M', q_0, \mathbf{A} \bigwedge_{i \in A} \Phi_i)$;

Hint for lower bound

We show a reduction from model checking LTL.

Complexity

SureWin is $PSPACE - complete$.

Decision problems: Nash Equilibrium (1)

Definition

Given a concurrent game G and a profile $s_{Ag} = (s_1, \ldots, s_i, \ldots, s_{|Ag|})$ of natural strategies under bound $k \in \mathbb{N}$:

s_{Ag} is a *Nash Equilibrium* in $G \Leftrightarrow \forall i \in Ag$, s_i is a best response.

Decision problems: Nash Equilibrium (1)

Definition

Given a concurrent game G and a profile $s_{Ag} = (s_1, \ldots, s_i, \ldots, s_{|Ag|})$ of natural strategies under bound $k \in \mathbb{N}$:

$$s_{Ag} \text{ is a } \textit{Nash Equilibrium} \text{ in } G \Leftrightarrow \forall i \in Ag, s_i \text{ is a best response.}$$

Best response

Given G, a player i, and a profile $s_{Ag} = (s_1, \ldots, s_i, \ldots, s_{|Ag|})$ under bound $k \in \mathbb{N}$, s_i is a *best response* in s_{Ag} if and only if:

$$path(s_{Ag}) \not\models \Phi_i \Rightarrow path((s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_{|Ag|})) \not\models \Phi_i$$

for all $s'_i \in \Sigma^r_i$ such that $compl(s'_i) \leq k$.

Decision problems: Nash Equilibrium (2)

Algorithm $\mathit{IsNotNash}(G, s_{Ag}, k)$:

for every $i \in Ag$ do
  if $\mathit{path}(s_{Ag}) \not\models \Phi_i$ then $s'_i = \mathit{GuessStrat}(G, i, k)$;
    if $\mathit{path}((s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_{|Ag|})) \models \Phi_i$ then return(true);
return (false);

Decision problems: Nash Equilibrium (2)

Algorithm $IsNotNash(G, s_{Ag}, k)$:

for every $i \in Ag$ do
  if $path(s_{Ag}) \not\models \Phi_i$ then $s'_i = GuessStrat(G, i, k)$;
    if $path((s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_{|Ag|})) \models \Phi_i$ then return(true);
return (false);

Hint for lower bound

We use a reduction from SAT.

Decision problems: Nash Equilibrium (2)

Algorithm $IsNotNash(G, s_{Ag}, k)$:

for every $i \in Ag$ do
if $path(s_{Ag}) \not\models \Phi_i$ then $s'_i = GuessStrat(G, i, k)$;
if $path((s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_{|Ag|})) \models \Phi_i$ then return(true);
return (false);

Hint for lower bound

We use a reduction from SAT.

Complexity

IsNotNash is $\mathbf{NP} - complete \Rightarrow$ *IsNash* is $co - \mathbf{NP} - complete$.

Decision problems: Nash Equilibrium (3)

Algorithm $\mathit{WinsSomeNash}(G, i, k)$:

$s_{Ag} = \mathit{GuessStrat}(G, Ag, k)$;
if $\mathit{path}(s_{Ag}) \models \Phi_i$ and not $\mathit{IsNotNash}(G, s_{Ag}, k)$ then return (true);
else return (false);

Decision problems: Nash Equilibrium (3)

Algorithm *WinsSomeNash*(G, i, k):

$s_{Ag} = GuessStrat(G, Ag, k)$;
if $path(s_{Ag}) \models \Phi_i$ and not $IsNotNash(G, s_{Ag}, k)$ then return (true);
else return (false);

Hint for lower bound

We use a reduction from QBF_2.

Decision problems: Nash Equilibrium (3)

Algorithm $WinsSomeNash(G, i, k)$:

$s_{Ag} = GuessStrat(G, Ag, k)$;
if $path(s_{Ag}) \models \Phi_i$ and not $IsNotNash(G, s_{Ag}, k)$ then return (true);
else return (false);

Hint for lower bound

We use a reduction from QBF_2.

Complexity

WinsSomeNash is $\mathbf{NP}^{\mathbf{NP}} = \Sigma_2^{\mathbf{P}}$-complete.

Decision problems: Nash Equilibrium (4)

Algorithm *LosesSomeNash*(G, i, k):

$s_{Ag} = GuessStrat(G, Ag, k)$;
if $path(s_{Ag}) \not\models \Phi_i$ and not $IsNotNash(G, s_{Ag}, k)$ then return (true);
else return (false);

Decision problems: Nash Equilibrium (4)

Algorithm *LosesSomeNash*(G, i, k):

$s_{Ag} = GuessStrat(G, Ag, k)$;
if $path(s_{Ag}) \not\models \Phi_i$ and not $IsNotNash(G, s_{Ag}, k)$ then return (true);
else return (false);

Hint for lower bound

We use a reduction from the complement problem of *WinsSomeNash*.

Decision problems: Nash Equilibrium (4)

Algorithm *LosesSomeNash*(G, i, k):

$s_{Ag} = GuessStrat(G, Ag, k)$;
if $path(s_{Ag}) \not\models \Phi_i$ and not $IsNotNash(G, s_{Ag}, k)$ then return (true);
else return (false);

Hint for lower bound

We use a reduction from the complement problem of *WinsSomeNash*.

Complexity

WinsAllNash is $co - \mathbf{NP}^{\mathbf{NP}} = \Pi_2^{\mathbf{P}}$-complete.

Conclusions

- We proposed the concept of *natural strategies*, based on an intuitive representation of conditional plans.
- We proposed how to measure the complexity of such strategies.
- We defined NatATL, a variant of alternating-time temporal logic to reason about natural strategic ability.
- We studied the complexity of NatATL model checking.
- We considered two main cases here: memoryless strategies and strategies with recall of the past.
- We showed that the relationship between natural strategies with recall and memoryless is more intricate than normally in *ATL*.
- We investigated some decision problems for natural abilities of agents in concurrent games with *LTL* winning conditions.

References I

R. Alur, T.A. Henzinger, and O. Kupferman.
Alternating-Time Temporal Logic.
Journal of ACM, 49(5):672–713, 2002.

E.M. Clarke and E.A. Emerson.
Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic.
In *LP'81*, pages 52–71, 1981.

W. Jamroga, V. Malvone, and A. Murano.
Natural strategic ability.
Artificial Intelligence (AIJ), (to appear).

W. Jamroga, V. Malvone, and A. Murano.
Reasoning about natural strategic ability.
In *AAMAS*, pages 714–722, 2017.

References II

F. Mogavero, A. Murano, and M.Y. Vardi.
Reasoning About Strategies.
In *FSTTCS'10*, pages 133–144, 2010.

A. Pnueli.
The Temporal Logic of Programs.
In *FOCS'77*, pages 46–57, 1977.

P.Y. Schobbens.
Alternating-Time Logic with Imperfect Recall.
ENTCS, 85(2):82–93, 2004.