Reasoning about Natural Strategic Ability

Vadim Malvone

Université d’Evry

December 7, 2018
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.
## 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.
There are three fundamental parts:

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

$M \models \varphi$
# Model Checking (2)

**Model**

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

- $\mathcal{AP}$ is a set of atomic propositions;
- $\mathcal{St}$ is a set of states;
- $S_0 \subseteq \mathcal{St}$ is a set of initial states;
- $R \subseteq \mathcal{St} \times \mathcal{St}$ is a transition relation;
- $L : \mathcal{St} \rightarrow 2^{\mathcal{AP}}$ 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 $\varphi$, the problem of the model checking consists to verify if $K, s \models \varphi$, for each initial state $s$. 
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.
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** ⇒ 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** ⇒ 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)**

<table>
<thead>
<tr>
<th>Plays and Histories</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The infinite paths starting from $s_I$ represent all possible plays.</td>
</tr>
<tr>
<td>• The finite paths starting from $s_I$ represent all possible histories.</td>
</tr>
<tr>
<td>• The set of all histories is denoted by $H$.</td>
</tr>
</tbody>
</table>
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 ($r$) $\Rightarrow \sigma : St \rightarrow Ac$;
  - memoryfull strategies ($R$) $\Rightarrow \sigma : H \rightarrow Ac$.
- 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**

- $\text{Ag} = \{\text{A: Alice, B: Bob}\}$
- $\text{Ac} = \{\text{P: Paper, R: Rock, S: Scissor}\}$
**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
**Example:** paper, rock, and scissors

- $\text{Ag} = \{A: \text{Alice, B: Bob}\}$
- $\text{Ac} = \{P: \text{Paper, R: Rock, S: Scissor}\}$
- $\text{St} = \{s_I, s_A, s_T, s_B\}$
- $s_I$: initial state
- $\text{AP} = \{w_A: \text{Alice wins, w_B: Bob wins}\}$
**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
- $AP = \{w_A: Alice wins, w_B: Bob wins\}$
- $DT = \{(P, P), (R, R), (S, S)\}$
**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)}**
- **D_A = {(P, R), (R, S), (S, P)}**
**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
- $AP = \{w_A: Alice wins, w_B: Bob wins\}$
- $DT = \{(P, P), (R, R), (S, S)\}$
- $DA = \{(P, R), (R, S), (S, P)\}$
- $DB = \{(R, P), (S, R), (P, S)\}$
**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)\}
- **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 $\sigma_A(h) = P$, $\forall h \in H$.
**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 $\sigma_A(h) = P$, $\forall h \in H$.
- A strategy for $B$ is $\sigma_B(h) = R$, $\forall h \in H$. 
**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 $\sigma_A(h) = P$, $\forall h \in H$.
- A strategy for $B$ is $\sigma_B(h) = R$, $\forall h \in H$.
- $\sigma_A$ and $\sigma_B$ induce the play $\pi$. 
Open Systems (4)

Specification

- **Internal** ⇒ directly over the game structure (Examples: Reachability, Safety, Nash Equilibrium).
- **External** ⇒ 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 $\langle A \rangle$ and $[A]$, with $A$ set of agents.
- The strategies are implicit.

Example: $\langle\{\alpha, \beta\}\rangle G \neg \text{fail}$: the agents $\alpha$ and $\beta$ have a collective strategy such that the system does not reach the failure state, independently from the strategies of the other agents.
### 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 $\langle A \rangle$ and $[A]$, with $A$ set of agents.
- The strategies are implicit.

### Example

$\langle \{\alpha, \beta\} \rangle \text{G}\neg \text{fail}$: the agents $\alpha$ and $\beta$ have a collective strategy such that the system does not reach the failure state, independently from the strategies of the other agents.
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].
• Strategic logics provide powerful tools to reason about multi-agent systems.
• However, strategies are *mathematical creatures*
Strategic logics provide powerful tools to reason about multi-agent systems.

However, strategies are mathematical creatures $\Rightarrow$ any function from system states to actions is fine.
• Strategic logics provide powerful tools to reason about multi-agent systems.
• However, strategies are mathematical creatures $\Rightarrow$ 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* $\Rightarrow$ *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.
A natural memoryless strategy $s_a$ for agent $a$ is a list of condition-action rules

$$(\text{cond}, \text{act})$$

such that:

- $\text{cond}$ is a boolean combination of propositions,
- $\text{act}$ is an available action in every state $q \models \text{cond}$,
- the last pair on the list is $(\top, \text{idle})$. 
Consider the following strategy for buying a train ticket:

1. $(\neg \text{ticket} \land \neg \text{selected}, \text{select})$;
2. $(\neg \text{ticket} \land \text{selected}, \text{pay})$;
3. $(\top, \text{idle})$. 
The complexity of strategy $s_a$ ($\text{compl}(s_a)$) can be defined by:

- Number of used propositions $\Rightarrow |\text{dom}(s_a)|$;
- 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:

$$\varphi ::= p | \neg \varphi | \varphi \land \varphi | \langle \langle A \rangle \rangle \leq^k X \varphi | \langle \langle A \rangle \rangle \leq^k \varphi U \varphi | \langle \langle A \rangle \rangle \leq^k \varphi W \varphi.$$ 

where $p \in AP$, $k \in \mathbb{N}$, and $A$ is a set of agents.
Reasoning about Natural Ability: NatATL

Syntax

A formula in NatATL is defined as:

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \langle A \rangle \leq^k X \varphi \mid \langle A \rangle \leq^k U \varphi \mid \langle A \rangle \leq^k W \varphi. \]

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

Semantics

\( M, q \models \langle A \rangle \leq^k \gamma \) iff there is a natural strategy \( s_A \) such that \( \text{compl}(s_A) \leq k \), and for each path \( \lambda \in \text{out}(q, s_A) \) 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 (∥⟨c⟩∥Fbuy).
• If the strategy is too complex, people won’t use it anyway.
• Instead, we should require ∥⟨c⟩∥≤kFbuy 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 ($\langle \langle c \rangle \rangle F_{buy}$).
- If the strategy is too complex, people won’t use it anyway.
- Instead, we should require $\langle \langle c \rangle \rangle \leq^k F_{buy}$ 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 $\langle \langle a \rangle \rangle \leq^k F_{win} \land \neg \langle \langle a \rangle \rangle \leq^{k-1} F_{win}$. 
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. \((\text{safe}^*, \text{digGold})\);
2. \((\text{safe}^* \cdot (\neg \text{safe} \land \text{haveGun}), \text{shoot})\);
3. \((\text{safe}^* \cdot (\neg \text{safe} \land \neg \text{haveGun}), \text{run})\);
4. \((\top^* \cdot (\neg \text{safe}) \cdot (\neg \text{safe}), \text{hide})\);
5. \((\top^*, \text{idle})\).
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:
   
   $M, q \models r \varphi$ 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:

   $M, q \models_R \varphi$ 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. \((\text{init}, \text{run})\);
2. \((\text{init} \cdot (\text{moved} \lor \text{stayed}), \text{run})));
3. \((\top^* \cdot \text{moved} \cdot \top, \text{lob})));
4. \((\top^* \cdot \text{stayed} \cdot \top, \text{shoot})));
5. \((\top^*, \text{idle})\).

The complexity of the strategy is 22.
Example: Soccer Scenario (3)

- Then, $\varphi = \langle 1 \rangle \leq 22 F_{\text{goal}}$ is true for strategies with recall.
- On the other hand, $\varphi$ 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 $\text{NatATL}_r$

- $\textbf{P}$ for fixed or bounded $k$;
- $\textbf{P}^{\text{NP}} = \Delta_2^\text{P}$-complete when $k$ is a parameter of the problem.
**Verification of Natural Strategies**

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

**Model checking $NatATL_R$**
- $\Delta_2^P$ for fixed or bounded $k$;
- $P^{NP^{NP}} = \Delta_3^P$ when $k$ is a parameter of the problem.
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 $M$,
- $\Phi : Ag \rightarrow L_{LTL}$ assigns each agent with an $LTL$ formula.
**Example: Simple market scenario**

- 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.
- $\Phi_{EC} = F_{\text{win}_1}$ and $\Phi_{NC} = F_{\text{win}_2}$.
**Definition**

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

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

Moreover, coalition $A$ surely wins in $G$ under bound $k$ iff it has a sure winning strategy of size at most $k$. 
Algorithm $\text{SureWin}(G, A, k)$:

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

**Decision problems:** Surely Winning (2)
Decision problems: Surely Winning (2)

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

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

Hint for lower bound

We show a reduction from model checking LTL.
**Algorithm** SureWin\((G, A, k)\):

\[
s_A = \text{GuessStrat}(G, A, k);
\]

Prune \(M\) according to \(s_A\), obtaining model \(M'\);

\[
\text{return } m\text{Check}_{\text{CTL}^*}(M', q_0, A \land_{i \in A} \Phi_i);
\]

**Hint for lower bound**

We show a reduction from model checking LTL.

**Complexity**

*SureWin* is \(PSPACE – complete\).
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 $G$ if and only if:

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

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

for all $s'_i \in \Sigma'_i$ such that $\text{compl}(s'_i) \leq k$. 
Algorithm $\text{IsNotNash}(G, s_{Ag}, k)$:

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

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

Hint for lower bound

We use a reduction from $\text{SAT}$. 
**Decision problems: Nash Equilibrium (2)**

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

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

**Hint for lower bound**

We use a reduction from $\text{SAT}$.

**Complexity**

$\text{IsNotNash}$ is $\textbf{NP} – \text{ complete } \Rightarrow \text{IsNash}$ is $\textbf{co-NP} – \text{ complete}$. 
Algorithm WinsSomeNash\((G, i, k)\):

\[
\begin{align*}
    s_{Ag} &= \text{GuessStrat}(G, Ag, k); \\
    \text{if } \text{path}(s_{Ag}) \models \Phi_i \text{ and not NotNash}(G, s_{Ag}, k) \text{ then return (true)}; \\
    \text{else return (false)};
\end{align*}
\]
Algorithm $\text{WinsSomeNash}(G, i, k)$:

$$s_{Ag} = \text{GuessStrat}(G, Ag, k);$$

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

Hint for lower bound

We use a reduction from $QBF_2$. 
Algorithm $\text{WinsSomeNash}(G, i, k)$:

\[
s_{Ag} = \text{GuessStrat}(G, Ag, k);
\]

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

Hint for lower bound

We use a reduction from $QBF_2$.

Complexity

$\text{WinsSomeNash}$ is $\mathbf{NP}^{\mathbf{NP}} = \Sigma_2^P$-complete.
Decision problems: Nash Equilibrium (4)

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

\[ s_{Ag} = \text{GuessStrat}(G, Ag, k); \]

if \( \text{path}(s_{Ag}) \not\subseteq \Phi_i \) and not *IsNotNash*(G, s_{Ag}, k) then return (true);
else return (false);
**Algorithm** \( \text{LosesSomeNash}(G, i, k) \):

\[
s_{Ag} = \text{GuessStrat}(G, Ag, k);
\]

if \( \text{path}(s_{Ag}) \not\subseteq \Phi_i \) and not \( \text{IsNotNash}(G, s_{Ag}, k) \) then return (true); else return (false);

**Hint for lower bound**

We use a reduction from the complement problem of \( \text{WinsSomeNash} \).
Algorithm \textit{LosesSomeNash}(G, i, k):

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

Hint for lower bound
We use a reduction from the complement problem of \textit{WinsSomeNash}.

Complexity
\textit{WinsAllNash} is co-\text{NP} = \Pi^p_2\text{-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 II

