Computationally Feasible Strategies

Catalin Dima[1] and Wojtek Jamroga[2]

[1] LACL, Université Paris Est Créteil

[2] University of Luxembourg & Polish Academy of Sciences

MAFTEC, 2025

Outline

1. Introduction and motivation
2. Computational Strategies
3. Computationally Bounded Ability
4. Main Results
5. Conclusions and future work

Outline

1 Introduction and motivation

2 Computational Strategies

3 Computationally Bounded Ability

4 Main Results

5 Conclusions and future work

Specification and Verification of Strategic Ability

- Many important properties are based on **strategic ability**:
  - Programs implementing desired agent behaviors.
  - Controllers in a cyber-physical system.
  - Plans for attackers against some systems.

- One can formalize such properties in logics of strategic ability, such as ATL or Strategy Logic
- ...and verify them by model checking

Limited computational power

- Strategies/plans might be hard to implement/follow by some agents.
- Objectives might need to be achieved in limited time.
- Strategies might need to be computed/applicable to more than one single system.

Rudiments of strategy complexity?

Motivation: Cryptographic Security

EAV Security, CPA Security

A cryptographic protocol is insecure if the cipher can be compromised with non-negligible probability by an adversary whose **strategy** is implemented as a **probabilistic Turing machine running in polynomial time**.

The protocol depends on some security parameters.

Motivation: Cryptographic Security

EAV Security, CPA Security

A cryptographic protocol is insecure if the cipher can be compromised ~~with non-negligible probability~~ by an adversary whose **strategy** is implemented as a ~~**probabilistic**~~ **Turing machine running in polynomial time**.

The protocol depends on some security parameters.

Motivation: Cryptographic Security

EAV Security, CPA Security

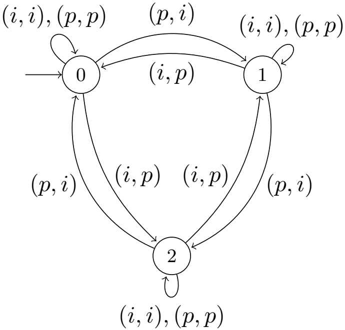
A cryptographic protocol is insecure if the cipher can be compromised ~~with non-negligible probability~~ by an adversary whose **strategy** is implemented as a **~~probabilistic~~ Turing machine running in ~~polynomial time~~ complexity class $\mathcal{C}$**.

The protocol depends on some security parameters.

Outline

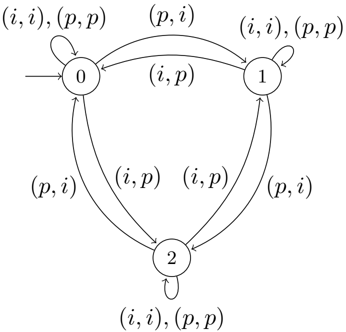
1. Introduction and motivation
2. Computational Strategies
3. Computationally Bounded Ability
4. Main Results
5. Conclusions and future work

Concurrent Game Structures, aka CGS



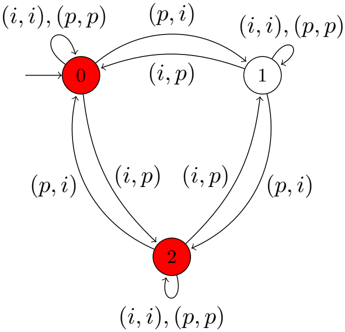
- Agents act simultaneously.
- Choice between two actions: push (p) or idle (i).
- Objective for agent $alice$: avoid state 2.
  - Push in state 0, idle in state 1, anything in state 2 (just for completion!).

Concurrent Game Structures, aka CGS



- Agents act simultaneously.
- Choice between two actions: push (p) or idle (i).
- Objective for both agents: avoid state 2.
  - Both idle in each state.

CGS with imperfect information, aka iCGS



- Observation is blurred in states 0 or 2 for agent $alice$.
- She can only exert the same action in both states.
- Still, she has a winning strategy: push in state 0 or 2, idle in state 1.
  - Knowledge of initial state is important.

Computational Strategies

Definition 1 (Computational strategy)

A **computational strategy** s for agent a in model M is an input/output Turing machine that takes as input a sequence of a's observations, and returns as output an action for a.

Computational Strategies

Definition 1 (Computational strategy)

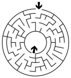
A **computational strategy** s for agent a in model M is an input/output Turing machine that takes as input a sequence of a's observations, and returns as output an action for a.

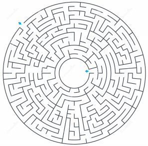
But such a strategy can only be applied to one model...

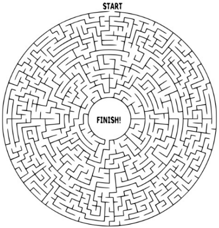
Model Templates

Definition 2 (Model template)

A **model template** is a countable family of concurrent game structures $\mathcal{M} = (M_1, M_2, \ldots)$.





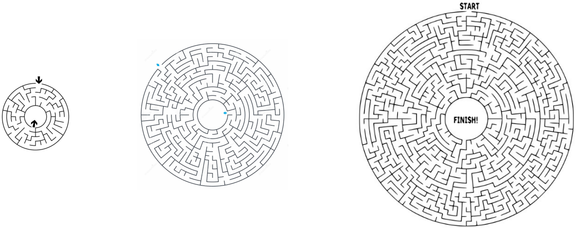


...

Model Templates

Definition 2 (Model template)

A **model template** is a countable family of concurrent game structures $\mathcal{M} = (M_1, M_2, \ldots)$.



...

All models share the same set of atomic propositions AP.

Some examples of model templates

- The familiy of mazes.

Some examples of model templates

- The familiy of mazes.
- A security protocol, depending on:
  - Nonces.
  - Security parameters.
  - Roles.

Some examples of model templates

- The familiy of mazes.
- A security protocol, depending on:
  - Nonces.
  - Security parameters.
  - Roles.
- A voting protocol:
  - Voters, candidates.
  - Security parameters.
- Etc.

Strategy Templates

Definition 3 (General computational strategy)

A **general computational strategy** S is an input/output Turing machine with 2 input tapes and 1 output tape that takes as input a model and a sequence of a's observations, and returns as output an action for a.

Strategy Templates

Example 1: getting out of any maze:

- Input tape 1: actual maze.
- Input tape 2: position in the maze.
- Output: next action.

Strategy Templates

Example 1: getting out of any maze:

- Input tape 1: actual maze.
- Input tape 2: position in the maze.
- Output: next action.

Example 2: an attack on a security protocols:

- Input tape 1: protocol instance generated by some security parameters/role assignment/nonce values etc.
- Input tape 2: current history.
- Output tape: current action of the attacker.

Outline

1. Introduction and motivation
2. Computational Strategies
3. Computationally Bounded Ability
4. Main Results
5. Conclusions and future work

Uniform Computational Ability

Definition 4 (Uniform computational ability)

Let $\mathcal{M}$ be a model template, φ an $\mathbf{LTL}$ objective in $\mathcal{M}$, and $\mathcal{C}$ a complexity class.

Agents $A \subseteq \mathbb{A}\mathrm{gt}$ have **uniform $\mathcal{C}$-ability** in $\mathcal{M}$ for φ, denoted:

$$\mathcal{M}, A \models_{\mathcal{C}} \varphi$$

if there exists a general strategy $\boldsymbol{S}_A$ for A, such that:

1. For every path $\lambda \in out(\mathcal{M}, \boldsymbol{S}_A)$, we have that $\lambda \models_{\mathbf{LTL}} \varphi$, and
2. There exists $f \in \mathcal{C}$ such that $\forall n, i \,.\, time_{\boldsymbol{S}_A}(n, i) \leq f(n, i)$.

$time_{\boldsymbol{S}_A}(n, i) =$ the maximal time taken by the TM $\boldsymbol{S}_A$ on model $\mathcal{M}(n)$ and observation sequences of length i.

Adaptive Computational Ability

Definition 5 (Adaptive computational ability)

Agents $A \subseteq \mathbb{A}\mathrm{gt}$ have **adaptive $\mathcal{C}$-ability** in $\mathcal{M}$ for φ, denoted:

$$\mathcal{M}, A \models_{\mathcal{C}} \varphi$$

if there exists a family of computational strategies $ST = (ST_M)_{M \in \mathcal{M}}$, $ST_M = (ST_{M,a})_{a \in A}$, such that:

1. For every n and path $\lambda \in out(M_n, ST_{M_n})$, we have that $\lambda \models_{\mathbf{LTL}} \varphi$, and
2. There exists $f \in \mathcal{C}$ such that $\forall n, i \,.\, time_{ST}(n, i) \leq f(n, i)$.

$time_{ST}(n, i) =$ the maximal time taken by the TM ST_{M_n} on observation sequences of length i.

Outline

1 Introduction and motivation

2 Computational Strategies

3 Computationally Bounded Ability

4 Main Results

5 Conclusions and future work

Hierarchy of Abilities

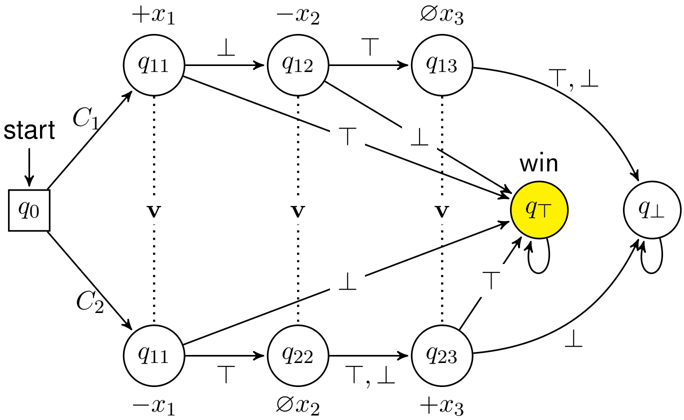
Theorem 6 (Hierarchy of abilities)

There exists a model template $\mathcal{M}$, coalition A in $\mathcal{M}$, and $\mathbf{LTL}$ objective φ, such that $\mathcal{M}, A \models_{\mathbf{EXPTIME}} \varphi$ but **not** $\mathcal{M}, A \models_{\mathbf{P}} \varphi$.

So, computational abilities form a proper hierarchy.

Proof idea: use an encoding of SAT as a model template.

Hierarchy of Abilities



iCGS M_ϕ for $\phi \equiv (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$.

Hierarchy of Abilities

- $\mathcal{M}_{Sat}$ = games M_ϕ for satisfiable formulas.
- $\mathcal{M}_{Unsat}$ = games M_ϕ for unsatisfiable formulas.
- Note that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{EXPTIME}} \Diamond$win.

Hierarchy of Abilities

- $\mathcal{M}_{Sat}$ = games M_ϕ for satisfiable formulas.
- $\mathcal{M}_{Unsat}$ = games M_ϕ for unsatisfiable formulas.
- Note that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{EXPTIME}} \Diamond$win.

- Suppose that $\mathcal{M}_{Sat}, \{\mathbf{v}\} \models_{\mathbf{P}} \Diamond$win.
  - I.e., verifier has a polynomial-time general strategy $\boldsymbol{S}_{\mathbf{v}}$ that obtains $\Diamond$win in $\mathcal{M}_{Sat}$
- From $\boldsymbol{S}_{\mathbf{v}}$ build a deterministic polynomial-time algorithm to solve SAT:
  1. Given a Boolean formula ϕ, construct M_ϕ.
  2. Generate the prefixes $\leq k+2$ for all the runs of $\boldsymbol{S}_{\mathbf{v}}$ in M_ϕ.
  3. If all prefixes end up in $q_\top$, return <u>true</u>; otherwise, return <u>false</u>.

Uniform $\subsetneq$ Adaptive

Theorem 7 (Uniform $\subseteq$ Adaptative)

If $\mathcal{M}, A \models\!\!\!\equiv_{\mathcal{C}} \varphi$ *then* $\mathcal{M}, A \models_{\mathcal{C}} \varphi$*.*

Uniform $\subsetneq$ Adaptive

Theorem 7 (Uniform $\subseteq$ Adaptative)

If $\mathcal{M}, A \mathrel{|\!\!\equiv}_{\mathcal{C}} \varphi$ *then* $\mathcal{M}, A \models_{\mathcal{C}} \varphi$.

Theorem 8 (Uniform $\neq$ Adaptive)

There exists a model template $\mathcal{M}$*, coalition* A *in* $\mathcal{M}$*, and* $\mathbf{LTL}$ *objective* φ*, such that* $\mathcal{M}, A \models_{\mathbf{P}} \varphi$ *but* **not** $\mathcal{M}, A \mathrel{|\!\!\equiv}_{\mathbf{P}} \varphi$.

In other words, having a family of winning polynomial-time strategies, one for each game, **does not imply** that we have a general polynomial-time strategy to win them all.

Definition 9 (Model checking)

Input:

- A Turing machine gen which, given $k \in \mathbb{N}$ as input, generates M_k.
- An $\mathbf{LTL}$ formula φ.
- A coalition $A \subseteq \mathbb{A}\mathrm{gt}$.
- A complexity class $\mathcal{C}$.

Output: true if $\mathcal{M}, A \models_{\mathcal{C}} \varphi$, otherwise false.

Similar definition for adaptive abilities.

Bad news for model checking of computational ability

Theorem 10

Model checking for uniform computational abilities is undecidable for singleton coalitions with safety objectives and $\mathcal{C} = O(1)$*.*

Theorem 11

Model checking is undecidable for singleton model templates for coalitions of size 2 with safety objectives and polytime complexity.

Simple decidable cases

Theorem 12

Model checking for singleton families of games, singleton coalitions $A = \{a\}$, and complexity constraints from $O(n)$ up is decidable.

Theorem 13

Model checking computational abilities in multi-energy families of iCGS and singleton coalitions is decidable.

Outline

1 Introduction and motivation

2 Computational Strategies

3 Computationally Bounded Ability

4 Main Results

5 Conclusions and future work

Conclusions and future work

- Strict hierarchy of computational strategic ability.
- Some basic undecidability and decidability results for the model-checking problem.

Future work:

- Decidability of the model-checking problem for larger (parameterized) classes of iCGS with counters.
- Application to some security game analysis.
- The general strategy verification problem.


Questions?