Towards Practical, On-the-Fly Verification of Strategic Ability for Knowledge and Information Flow

Damian Kurpiewski, Mateusz Kamiński, Wojciech Jamroga

24/09/2025

Institute of Computer Science
Polish Academy of Sciences

Faculty of Mathematics and Computer Science
Nicolaus Copernicus University in Toruń

Model Checking of Strategic Abilities

ATL: What Agents Can Achieve

- ATL: Alternating-time Temporal Logic [Alur et al. 1997-2002]
- Temporal logic meets game theory
- Main idea: cooperation modalities

$\langle\langle A \rangle\rangle\Phi$: coalition A has a collective strategy to enforce Φ

$\rightsquigarrow$ Φ can include temporal operators: X (next), F (sometime in the future), G (always in the future), $\mathcal{U}$ (strong until)

Semantic Variants of ATL

Memory of agents:

- Perfect recall (R) vs. imperfect recall strategies (r)

Available information:

- Perfect information (I) vs. imperfect information strategies (i)

Example Formulae

- $\langle\langle \mathit{holmes} \rangle\rangle \mathrm{F}(\text{solve} \wedge \neg \text{falseAccus})$:
  "Sherlock Holmes can solve case without false accusation"

Example Formulae

- $\langle\langle \mathit{holmes} \rangle\rangle \mathrm{F}(\mathsf{solve} \wedge \neg \mathsf{falseAccus})$:
  "Sherlock Holmes can solve case without false accusation"

- $\langle\langle \mathit{holmes}, \mathit{watson} \rangle\rangle (\neg \mathsf{crisis})\, \mathrm{U}\, \mathsf{endOfStory}$:
  "Sherlock Holmes and Dr Watson are able to save Great Britain from the crisis until the end of the story"

ATL with incomplete information

- Imperfect information ($q \sim_a q'$)

ATL with incomplete information

- Imperfect information ($q \sim_a q'$)
- Imperfect recall - agent memory coded within state of the model

ATL with incomplete information

- Imperfect information ($q \sim_a q'$)
- Imperfect recall - agent memory coded within state of the model
- Uniform strategies - specify same choices for indistinguishable states:
  $q \sim_a q' \implies s_a(q) = s_a(q')$

ATL with incomplete information

- Imperfect information ($q \sim_a q'$)
- Imperfect recall - agent memory coded within state of the model
- Uniform strategies - specify same choices for indistinguishable states:
  $q \sim_a q' \implies s_a(q) = s_a(q')$
- Fixpoint equivalences **do not hold** anymore

ATL with incomplete information

- Imperfect information ($q \sim_a q'$)
- Imperfect recall - agent memory coded within state of the model
- Uniform strategies - specify same choices for indistinguishable states:
  $q \sim_a q' \implies s_a(q) = s_a(q')$
- Fixpoint equivalences **do not hold** anymore
- Model checking **ATL**$_{\mathrm{ir}}$ is Δ_2^p-complete

Formal Background

Modules

The main part of the input is given by a set of asynchronous modules, where local states are labelled with valuations of state variables

A module is a tuple $M = (S, Act, \delta, s_0, AP, L)$ where:

- S is a set of states.
- Act is a set of actions.
- $\delta : S \times Act \rightarrow S$ is the transition function.
- $s_0 \in S$ is the initial state.
- AP is a set of atomic propositions.
- $L : S \rightarrow 2^{AP}$ is the labeling function.

Strategies

Strategy

A strategy for agent a is a function $s_a : S \to Act$ such that for every state $q \in S$, $s_a(q)$ is an action available to a in q.

A strategy is uniform if for all states $q, q' \in S$, if $q \sim_a q'$, then $s_a(q) = s_a(q')$.

Outcome

The outcome of a strategy s_A for coalition A from state q is the set of all infinite paths $\lambda = q_0 q_1 q_2 \ldots$ such that $q_0 = q$ and for all $i \geq 0$, there exists a joint action α with $\alpha_a = s_a(q_i)$ for all $a \in A$ and $\delta(q_i, \alpha) = q_{i+1}$.

ATL$_{ir}$

Given a model M, a state q in the model, and a formula $\langle\langle A \rangle\rangle\varphi$, the formula holds in M, q iff there exists a uniform strategy s_A for coalition A such that for all states $q' \in S$ with $q \sim_A q'$, all paths in the outcome of s_A from q' satisfy φ.

ATLK$_{ir}$

Knowledge operator $K_a\varphi$

- Agent a knows that φ holds in all states indistinguishable to a.
- $K_a\varphi$ holds in M, q iff φ holds in all states q' such that $q \sim_a q'$.
- Useful for reasoning about what agents can deduce from their observations.

ATLH$_{ir}$

Hartley uncertainty operator $H_a^{\leq k}\{\psi_1, \ldots, \psi_n\}$

- Measures the uncertainty of agent a about a set of propositions.
- $H_a^{\leq k}$ means the agent's uncertainty is at most k bits.
- $H_a^{\leq k}\{\psi_1, \ldots, \psi_n\}$ holds in M, q iff the number of possible valuations for $\{\psi_1, \ldots, \psi_n\}$ in states indistinguishable to a from q can be represented on at most k bits.
- Applied to analyze information flow and privacy in multi-agent systems.

Example: Simple Model of Voting and Coercion

- Two agents: the Voter and the Coercer
- Two candidates

Example: Simple Model of Voting and Coercion

- Two agents: the Voter and the Coercer
- Two candidates

- Voter can cast her vote and then interact with the Coercer
- Voter can give (or not) her vote to the Coercer

Example: Simple Model of Voting and Coercion

- Two agents: the Voter and the Coercer
- Two candidates

- Voter can cast her vote and then interact with the Coercer
- Voter can give (or not) her vote to the Coercer

- Coercer can punish (or not) the voter

Example: Simple Model of Voting and Coercion

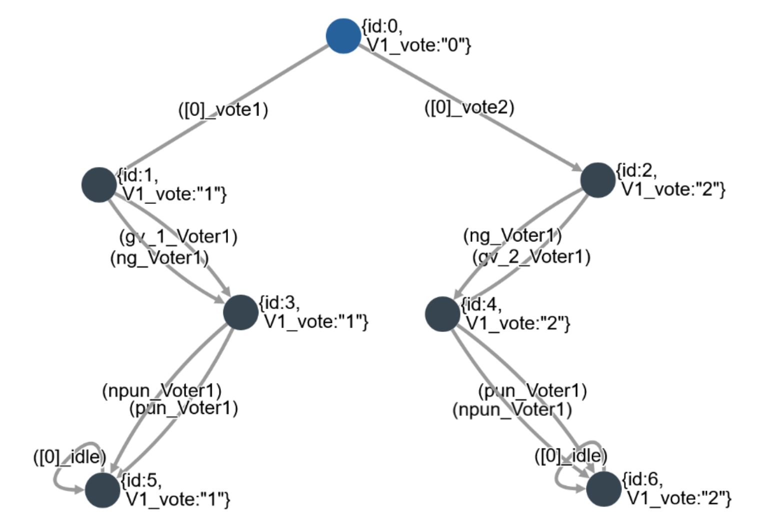
- Two agents: the Voter and the Coercer
- Two candidates

- Voter can cast her vote and then interact with the Coercer
- Voter can give (or not) her vote to the Coercer

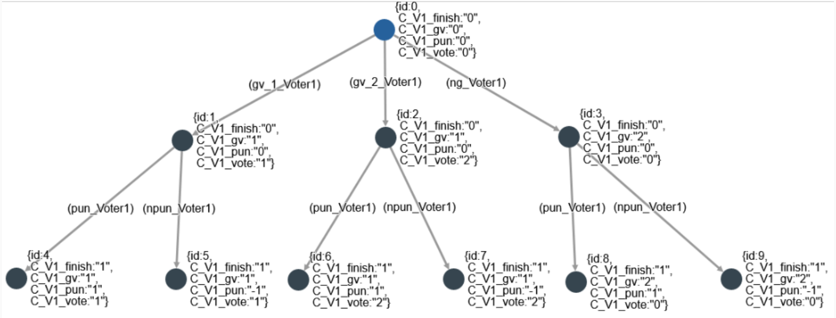
- Coercer can punish (or not) the voter

- Asynchronous semantics with synchronization over actions: vote giving and punishment are **synchronized**

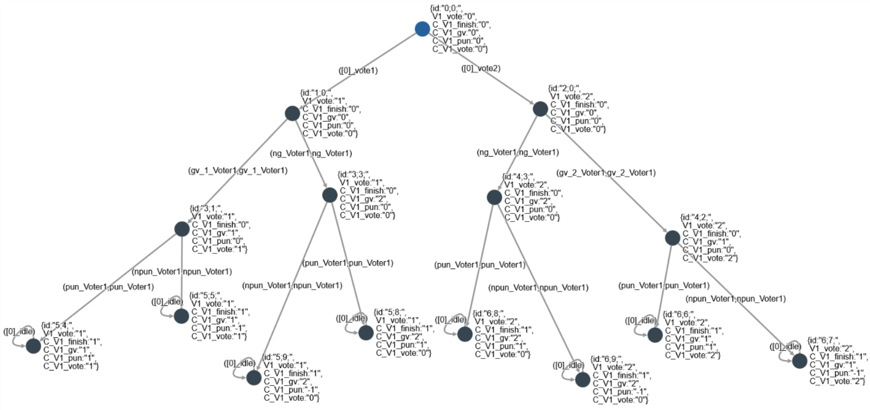
Example: Simple Model of Voting and Coercion
Voter Local Model



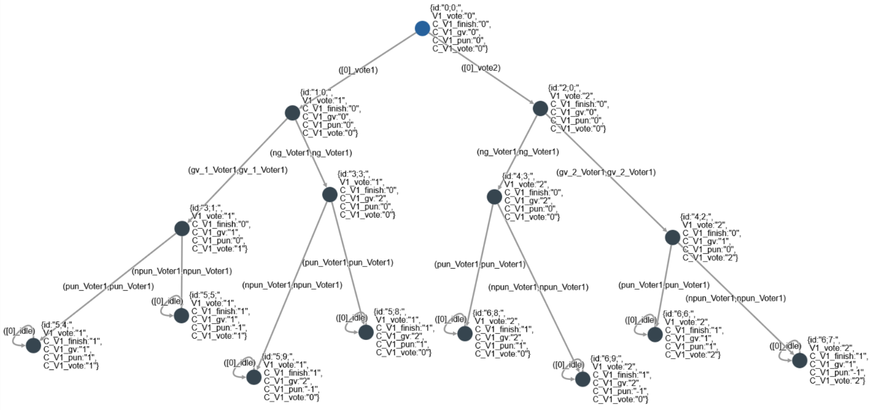
Example: Simple Model of Voting and Coercion
Coercer Local Model



Example: Simple Model of Voting and Coercion
Global Model



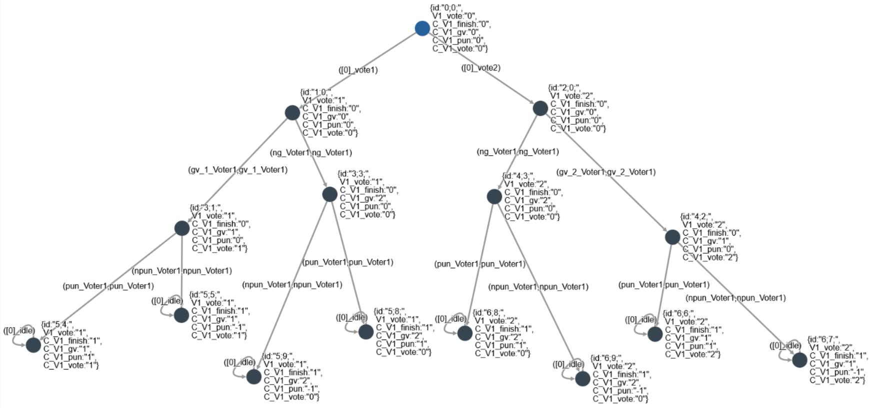
Example: Simple Model of Voting and Coercion
Example Formula



$\langle\langle \mathit{Coercer} \rangle\rangle \mathrm{F}\ \mathrm{pun}_1$:

"The Coercer can eventually punish the Voter"

Example: Simple Model of Voting and Coercion
Example Formula

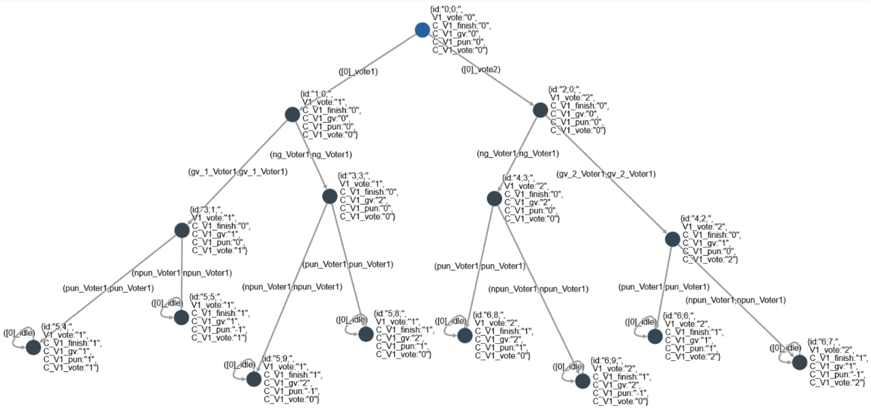


$\langle\langle \mathit{Coercer} \rangle\rangle \mathrm{F}\ \mathrm{pun}_1$:

"The Coercer can eventually punish the Voter"

TRUE

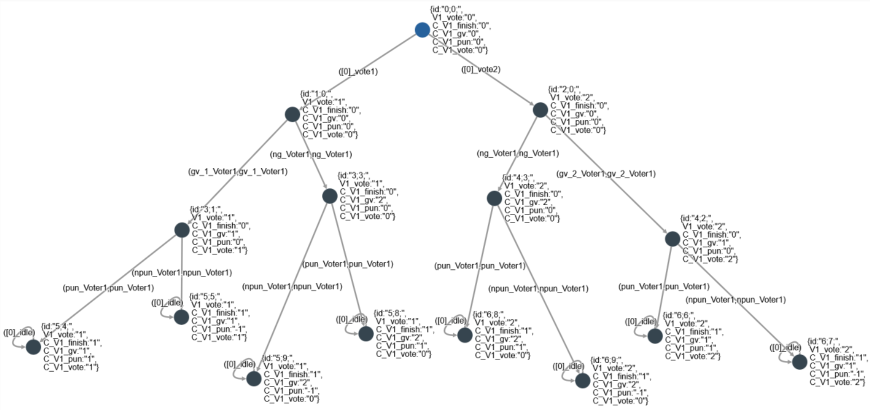
Example: Simple Model of Voting and Coercion
Example Formula



$\langle\langle \mathit{Coercer} \rangle\rangle \mathrm{G}(\mathsf{finish}_1 \wedge \mathsf{vote}_{1,1} \implies K_C \mathsf{vote}_{1,1})$:

"The Coercer knows when the Voter has voted for the first candidate"

Example: Simple Model of Voting and Coercion
Example Formula

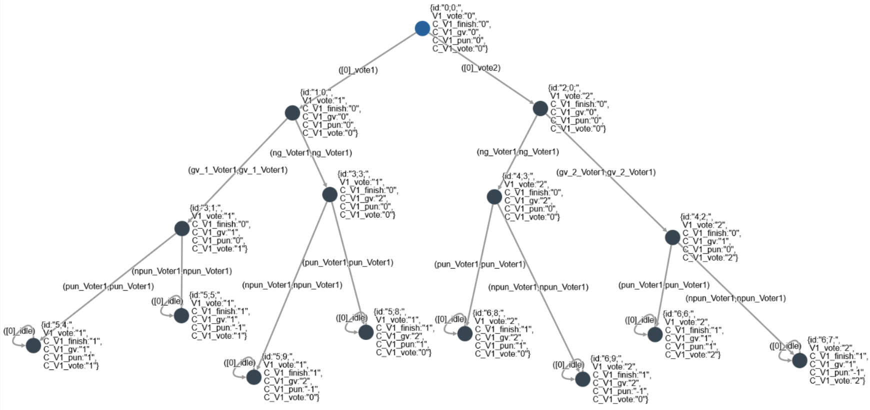


$\langle\langle Coercer \rangle\rangle \mathrm{G}(\mathrm{finish}_1 \wedge \mathrm{vote}_{1,1} \implies K_C \mathrm{vote}_{1,1})$:

"The Coercer knows when the Voter has voted for the first candidate"

FALSE

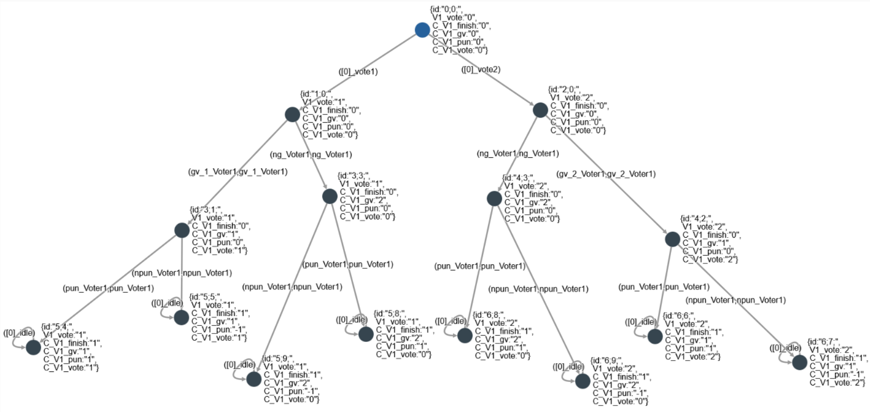
Example: Simple Model of Voting and Coercion
Example Formula



$\langle\langle \mathit{Coercer} \rangle\rangle \mathrm{G}(\mathrm{finish}_1 \implies H_C^{\leq 2}\{\mathrm{vote}_{1,1}, \mathrm{vote}_{1,2}\})$:

"The Coercer uncertainty about the Voter's vote is at most 2 bits"

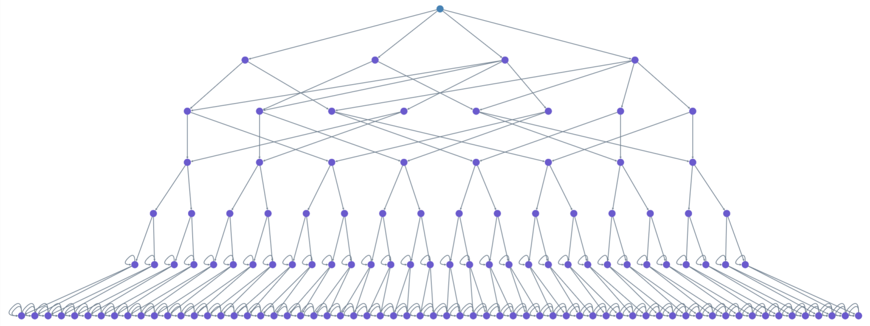
Example: Simple Model of Voting and Coercion
Example Formula



$\langle\langle \mathit{Coercer} \rangle\rangle \mathrm{G}(\mathrm{finish}_1 \implies H_C^{\leq 2}\{\mathrm{vote}_{1,1}, \mathrm{vote}_{1,2}\})$:
“The Coercer uncertainty about the Voter’s vote is at most 2 bits”

TRUE

Example: 2 Voters



Simple Specification Language

Simple Voting Model

```
Agent Voter1:
LOCAL: [V1_vote]
PERSISTENT: [V1_vote]
INITIAL: []
init q0
vote1: q0 -> q1 [V1_vote:=1]
vote2: q0 -> q1 [V1_vote:=2]
shared[2] gv_1_Voter1[gv_1_Voter1]: q1 [V1_vote==1] -> q2
shared[2] gv_2_Voter1[gv_1_Voter2]: q1 [V1_vote==2] -> q2
shared[2] ng_Voter1[ng_Voter1]: q1 -> q2
shared[2] pun_Voter1[pn_Voter1]: q2 -> q3
shared[2] npun_Voter1[pn_Voter1]: q2 -> q3
idle: q3 -> q3

FORMULA: <<Coercer>>[](C_V1_finish==0 ||
          (V1_vote==1 && &K_Coercer(V1_vote==1)) )
```

Agent

Initial configuration

Shared transition

Local name

Local transition

Guard

State (template)

Proposition variable

Formula

STV - Strategic Verifier

STV - Strategic Verifier

- Explicit-state model checking.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations, depth-first strategy synthesis and on-the-fly strategy synthesis.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations, depth-first strategy synthesis and on-the-fly strategy synthesis.
- Reduction methods: partial-order reductions and assume-guarantee reasoning.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations, depth-first strategy synthesis and on-the-fly strategy synthesis.
- Reduction methods: partial-order reductions and assume-guarantee reasoning.
- Asynchronous semantics with: action-oriented synchronization and data-oriented synchronization.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations, depth-first strategy synthesis and on-the-fly strategy synthesis.
- Reduction methods: partial-order reductions and assume-guarantee reasoning.
- Asynchronous semantics with: action-oriented synchronization and data-oriented synchronization.
- Properties: reachability and safety.

STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations, depth-first strategy synthesis and on-the-fly strategy synthesis.
- Reduction methods: partial-order reductions and assume-guarantee reasoning.
- Asynchronous semantics with: action-oriented synchronization and data-oriented synchronization.
- Properties: reachability and safety.
- Epistemic operators: knowledge and Hartley uncertainty.

Approximate Verification of Strategic Ability

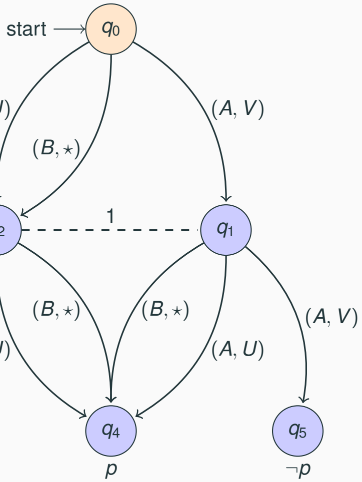
$M \models_{ir} \varphi$: **DIFFICULT!**

$$M \models LB(\varphi) \Rightarrow M \models_{ir} \varphi \Rightarrow M \models UB(\varphi)$$

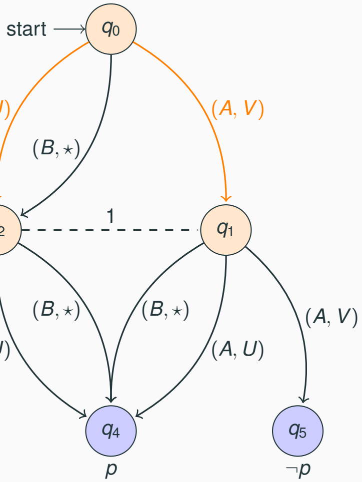
$\uparrow$ Alternating Epistemic Mu-Calculus

$\uparrow$ Perfect Information

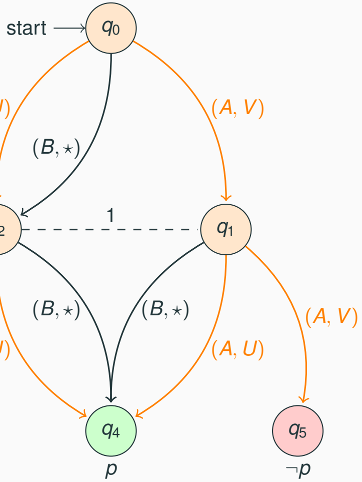
DFS Strategy Synthesis



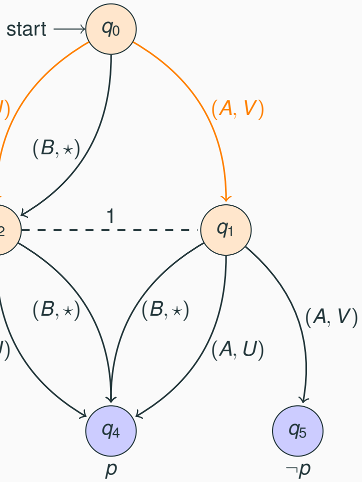
DFS Strategy Synthesis



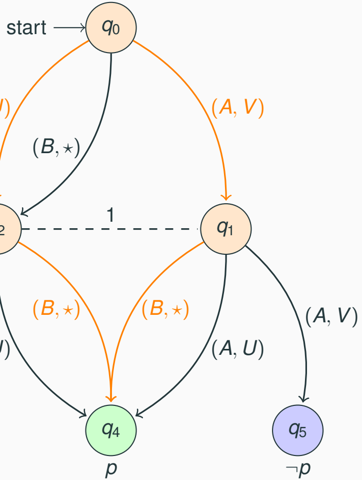
DFS Strategy Synthesis



DFS Strategy Synthesis

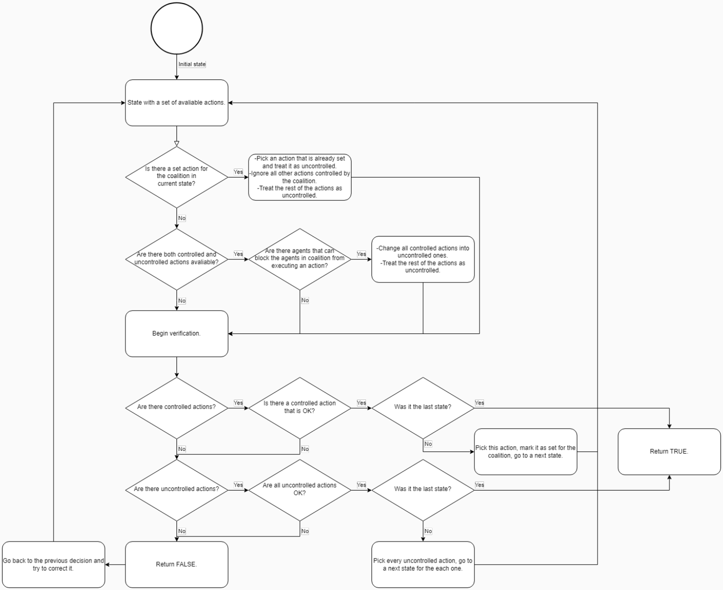


DFS Strategy Synthesis

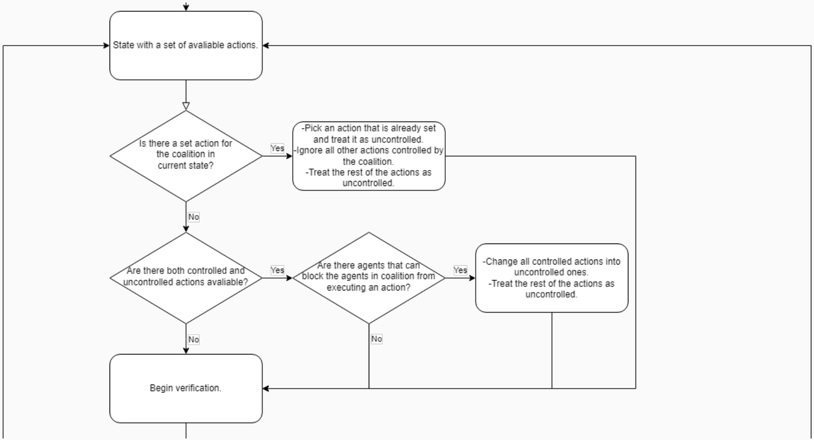


Challenges

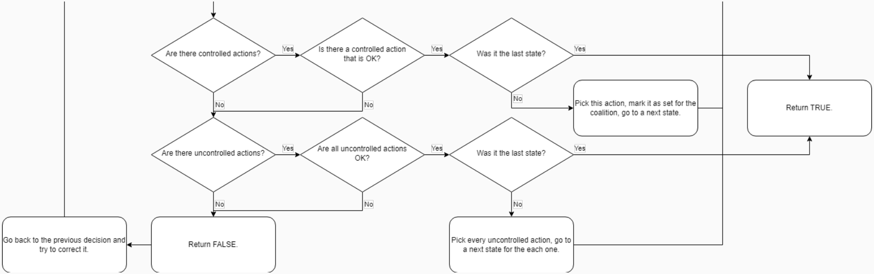
Algorithm Overview



Algorithm Part 1



Algorithm Part 2



Experimental Evaluation

Verification of Selene E-Voting Protocol

#A	Standard			On-the-fly		Res
	States	Gen	Verif	States	Verif	
4	3.85e4	3	<1	2.91e3	<1	True
5	2.19e6	179	<1	1.47e5	1	True
6	8.12e7	2642	<1	1.10e6	14	True
7	timeout			9.60e6	406	True
8	timeout					

Table 1: Results for ϕ_1 with 3 candidates and 3 revotes

$$\phi_1 \equiv \langle\langle C \rangle\rangle \mathrm{G}\big((\text{finish}_1 \wedge \text{revote} = 2 \wedge \text{voted}_1 = 1) \rightarrow K_C \text{voted}_1 = 1\big)$$

Conclusions

Conclusions

- Modal logics for MAS are characterized by high computational complexity.
- Verification of strategic properties in scenarios with imperfect information is **difficult**.
- Much complexity of model checking for strategic abilities is due to the size of the model of the system.
- STV addresses the challenge by implementing various **reduction** and **model-checking** methods which shows very promising performance.
- STV supports user-friendly modelling of MAS, and automated reduction and verification methods.
- Addition of knowledge and uncertainty operators allows verification of anonymity-related properties.

THANK YOU!