STV+AGR: Towards Verification of Strategic Ability using Assume-Guarantee Reasoning

Damian Kurpiewski

Institute of Computer Science, Polish Academy of Sciences
Nicolaus Copernicus University, Faculty of Mathematics and Computer Science

(joint work with Łukasz Mikulski and Wojtek Jamroga)

Valencia, 16/11/2020

Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

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), U (strong until)

Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

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 and depth-first strategy synthesis.

STV - Strategic Verifier

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

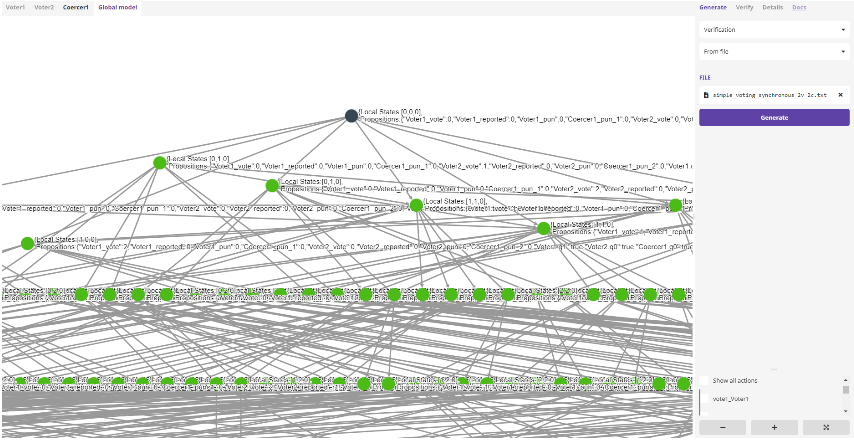
STV - Strategic Verifier

- Explicit-state model checking.
- User-defined input.
- Web-based graphical interface.
- Model-checking algorithms: fixpoint-approximations and depth-first 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 and depth-first 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.

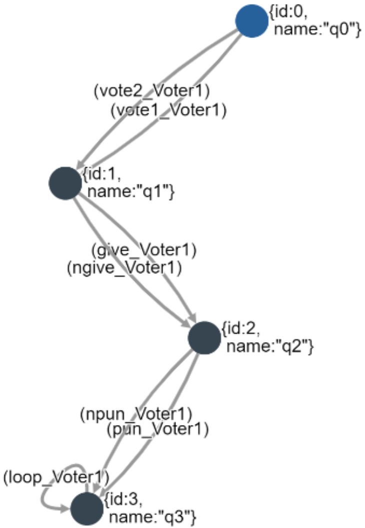
Simple Voting model in STV



Model specification - voter

```
Agent Voter[2]:

init: q0

vote1: q0 -> q1 [aID_vote=1]
vote2: q0 -> q1 [aID_vote=2]

give: q1 -> q2 [aID_reported=?aID_vote]
ngive: q1 -> q2 [aID_reported=-1]

pun: q2 -[Coercer1_pun_ID == 1]> q3 [aID_pun=1]
npun: q2 -[Coercer1_pun_ID == -1]> q3 [aID_pun=0]

loop: q3->q3
```

Voter model in STV



Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

Automated assumptions

1. Design the model and create a specification file.

Automated assumptions

1. Design the model and create a specification file.
2. Split the agents into assumption groups.

Automated assumptions

1. Design the model and create a specification file.
2. Split the agents into assumption groups.
3. Each assumption group should specify the coalition and the formula. Environment group should not specify the formula.

Automated assumptions

1. Design the model and create a specification file.
2. Split the agents into assumption groups.
3. Each assumption group should specify the coalition and the formula.
   Environment group should not specify the formula.
4. Use STV to automatically generate specification files for each assumption group.

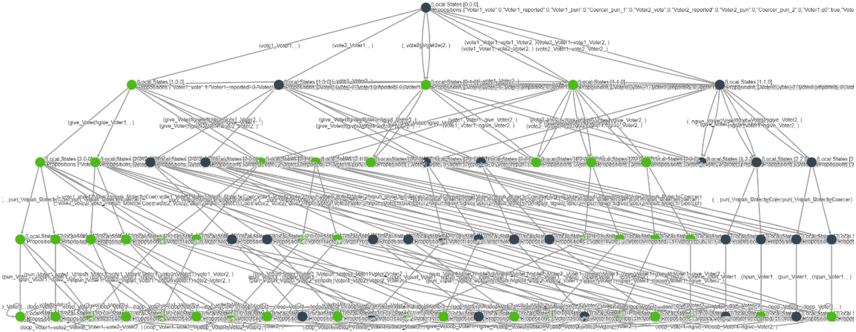
Automated assumptions

1. Design the model and create a specification file.
2. Split the agents into assumption groups.
3. Each assumption group should specify the coalition and the formula. Environment group should not specify the formula.
4. Use STV to automatically generate specification files for each assumption group.
5. Verify each model in the tool.

Assumption specification

```
Group 1:

Agent Voter1:
init: q0
LOCAL: [Voter1_vote, Voter1_reported, Voter1_pun]
INTERFACE: [Coercer_pun_1]
vote1: q0 -> q1 [Voter1_vote=1]
vote2: q0 -> q1 [Voter1_vote=2]
give: q1 -> q2 [Voter1_reported=?Voter1_vote]
ngive: q1 -> q2 [Voter1_reported=-1]
pun: q2 -[Coercer_pun_1 == 1]> q3 [Voter1_pun=1]
npun: q2 -[Coercer_pun_1 == -1]> q3 [Voter1_pun=-1]
loop: q3->q3

FORMULA: <<Voter1>>G(Voter1_pun=-1 | Voter1_vote=1)
```

Model generated from the assumption



Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

Simple Voting Model: Results

Formula

$$\langle\langle \mathit{Voter}_1 \rangle\rangle \mathrm{G}(\neg \mathrm{pstatus}_1 \vee \mathrm{voted}_1 = 1)$$

V	Monolithic model checking				Assume-guarantee verification			
	#st	**#tr**	**DFS**	**Apprx**	**#st**	**#tr**	**DFS**	**Apprx**
2	529	2216	< 0.1	$< 0.1/$*Yes*	161	528	< 0.1	$< 0.1/$*Yes*
3	12167	127558	< 0.1	0.8/*Yes*	1127	7830	< 0.1	$< 0.1/$*Yes*
4	2.79*e*5	6.73*e*6	memout		7889	1.08*e*5	< 0.1	0.8/*Yes*
5	memout				5.52*e*4	1.45*e*6	< 0.1	11/*Yes*

Table: Results of assume-guarantee verification (times given in seconds)

Outline

Logical Specification of Strategic Abilities

Tool

Assume-guarantee verification

Evaluation

Conclusions

Conclusions

- 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 a compositional model checking scheme, called assume-guarantee verification.
- STV supports user-friendly modelling of MAS, and automated generation of abstractions that are used as assumptions in the scheme.

THANK YOU