Verification of Multi-Agent Properties in Electronic Voting: A Case Study

Damian Kurpiewski, Wojciech Jamroga, Łukasz Maśko, Łukasz Mikulski, Wojciech Penczek, Teofil Sidoruk

The problem

- Verification of strategic abilities under **imperfect information**
- Logic: $\mathbf{ATL_{ir}}$
- Complexity: $\Delta_2^P - \text{complete}$

Simple Voting Model

Example

Agents

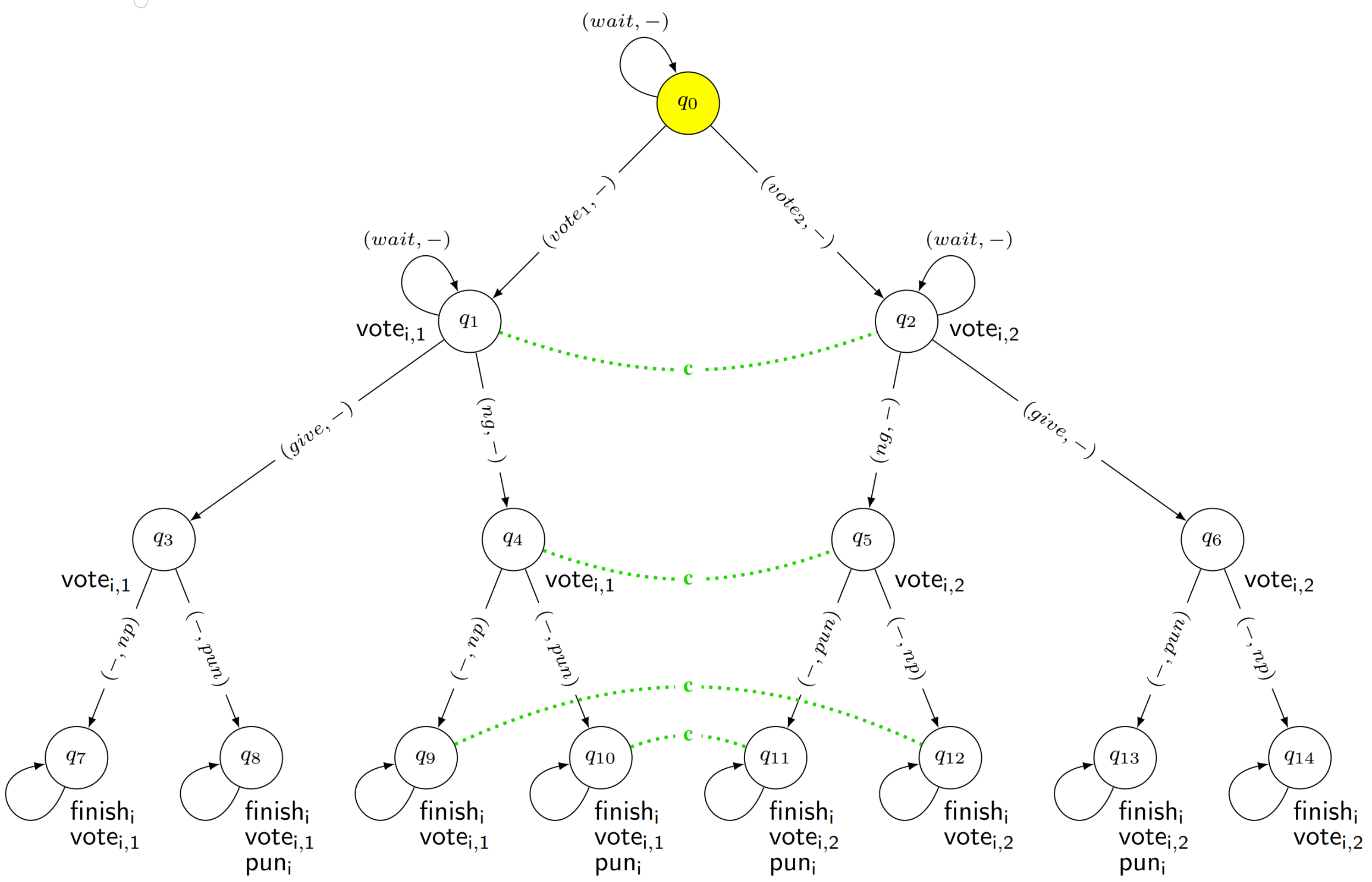
Voter

- Casts her vote
- Decides to show (or not) her vote to the Coercer

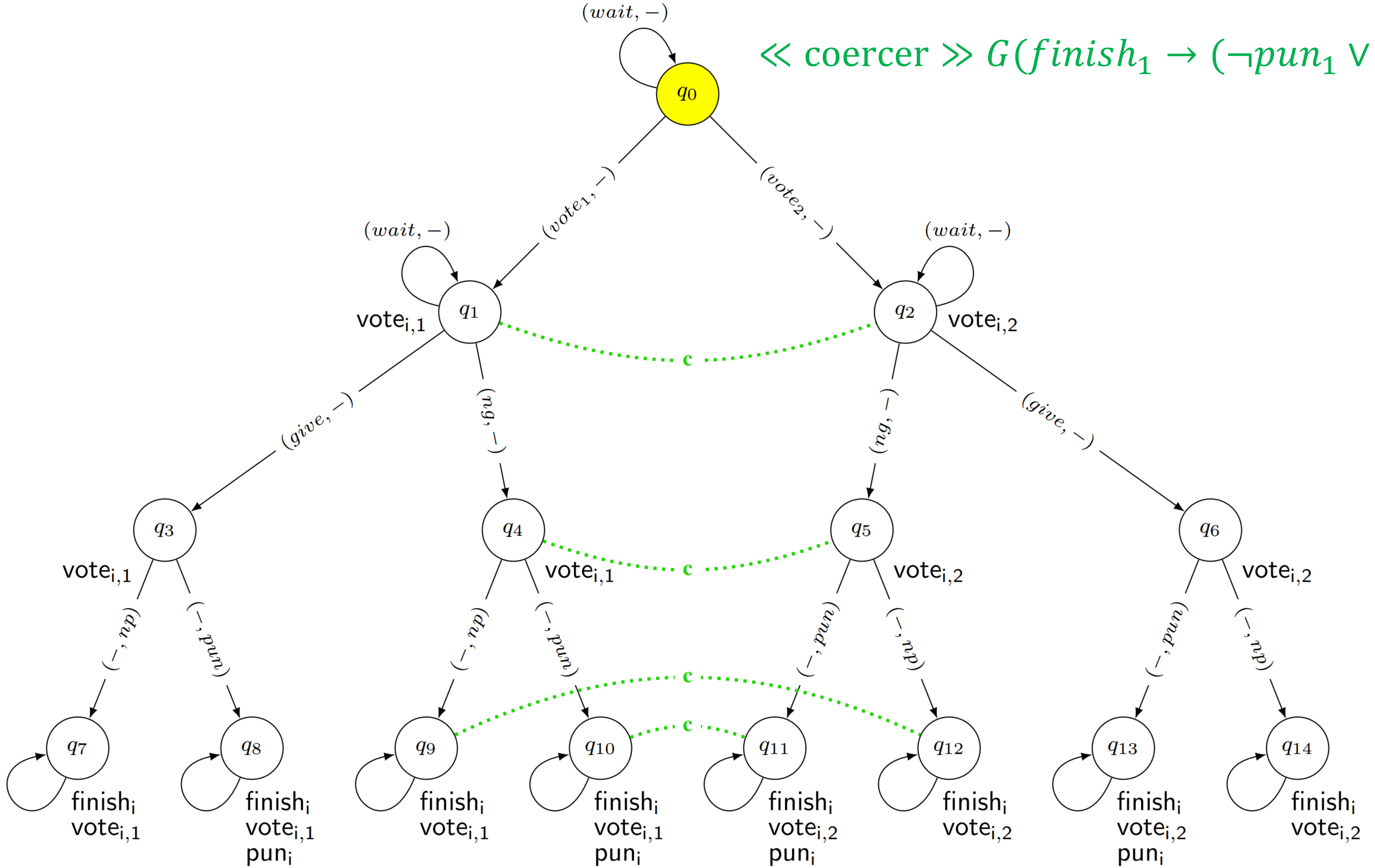
Coercer

- Gets (or not) the vote from the Voter
- Decides to punish (or not) the Voter

1 Voter
1 Coercer
2 Candidates

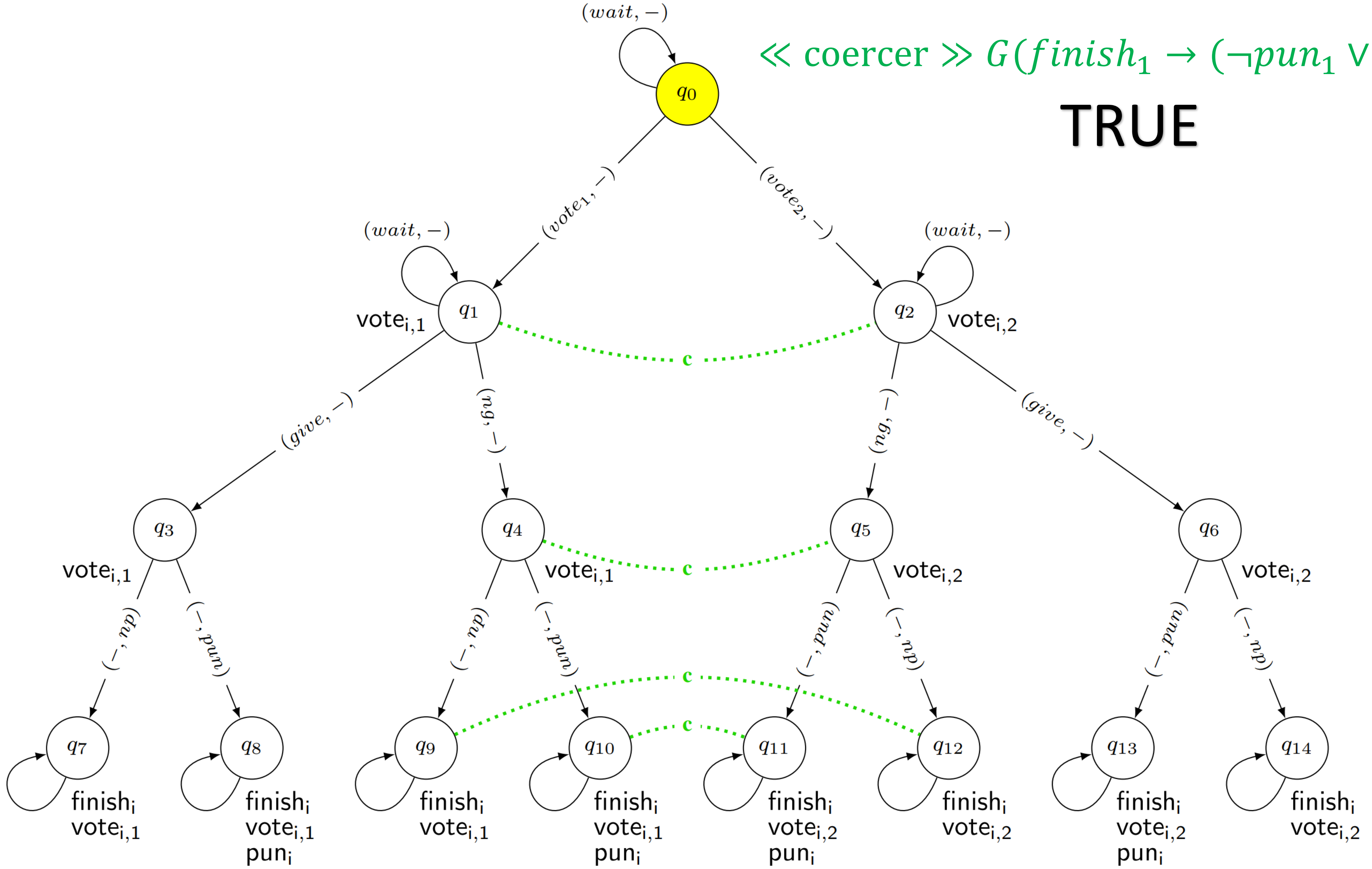


$\ll \text{coercer} \gg G(finish_1 \rightarrow (\neg pun_1 \lor vote_{1,1}))$

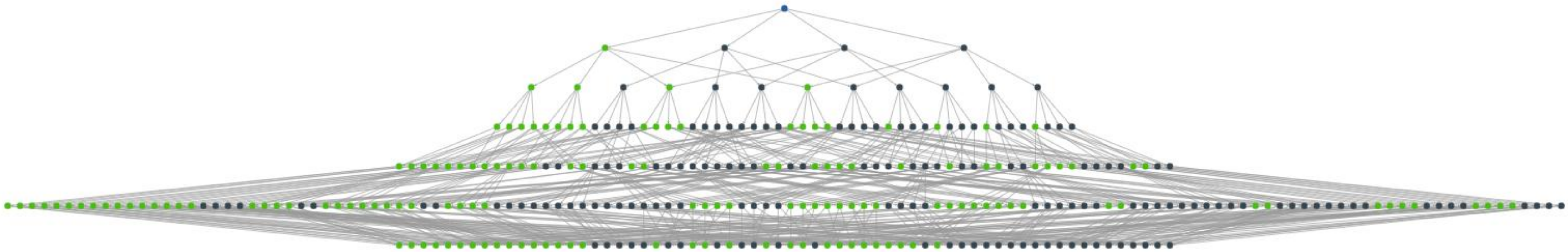


$\ll \text{coercer} \gg G(finish_1 \rightarrow (\neg pun_1 \vee vote_{1,1}))$

TRUE



2 Voters, 1 Coercer, 2 Candidates



The solution(?)

- Fixpoint approximations
- DFS and DominoDFS strategy synthesis
- Parallel DFS strategy synthesis
- Partial-order reductions

Fixpoint approximations

- Fixpoint computation is (usually) efficient
- Fixpoint equivalences do not hold for $\mathrm{ATL_{ir}}$
- **Lower bound**: translation to $\mathrm{AE}\mu\mathrm{C}$
- **Upper bound**: $\mathrm{ATL_{Ir}}$ (perfect information)
- Sometimes bounds don’t match

DFS strategy synthesis

- Recursive search from the initial state
- Synthesize winning strategy during the search
- Better than exhaustive search through the entire strategy space
- Handling epistemic classes can be troublesome

DominoDFS strategy synthesis

- DFS + domination relations
- Observation: some strategies **dominate** others
- Dominated strategies can be omitted during the search

Parallel DFS strategy synthesis

- Main problems to consider:
  - It is difficult (if not impossible) to split the model data between processes
  - Epistemic classes can join states in different parts of the model
  - Backtracing is not as simple as it seems
- Several different approaches to parallelization
- Best promising approach:
  - Split the work early (preferably from the initial state)
  - Each proces has own copy of the whole model
  - Split by agent-controlled transitions

Partial-order reductions

- Asynchronous models
- State-space explosion related to interlacing
- Effective reduction methods exists for LTL and can be adapted to $\mathrm{ATL_{ir}}$

Selene e-voting Protocol Model

Case Study

Agents

Election Authority

- Generates trackers and sends them to voters
- Collects votes and publishes them
- Serves as the Web Bulletin Board

Voter

- Casts her vote
- Uses tracker to check the WBB

Coerced Voter

- Interacts with the Coercer
- Casts her vote
- Can generate false tracker and show it to the Coercer

Coercer

- Interacts only with coerced voters, telling them the desired vote
- Gathers trackers from coerced voters
- Can punish coerced voters

Re-voting scheme

Coerced voter can vote several times

Each vote, apart from the last one, is shared with the coercer

Last vote (if cast) **is private**

Coerced Voter (3 candidates, 3 revotes)

```
Agent VoterC[1]:
init start
shared coerce1_aID: start -> coerced [aID_required=1]
shared coerce2_aID: start -> coerced [aID_required=2]
shared coerce3_aID: start -> coerced [aID_required=3]
select_vote1: coerced -> prepared [aID_vote=1, aID_prep_vote=1]
select_vote2: coerced -> prepared [aID_vote=2, aID_prep_vote=2]
select_vote3: coerced -> prepared [aID_vote=3, aID_prep_vote=3]
shared is_ready: prepared -> ready
shared start_voting: ready -> voting
shared aID_vote: voting -> vote [Coercer1_aID_vote=?aID_vote, Coercer1_aID_revote=?aID_revote]
shared send_vote_aID: vote -> send
revote_vote_1: send -[aID_revote==1]> voting [aID_vote=?aID_required, aID_revote=2]
skip_revote_1: send -[aID_revote==1]> votingf
revote_vote_2: send -[aID_revote==2]> voting [aID_vote=?aID_required, aID_revote=3]
skip_revote_2: send -[aID_revote==2]> votingf
final_vote: send -[aID_revote==3]> votingf [aID_vote=?aID_prep_vote]
skip_final: send -[aID_revote==3]> votingf
shared send_fvote_aID: votingf -> sendf
shared finish_voting: sendf -> finish
shared send_tracker_aID: finish -> tracker
shared finish_sending_trackers: tracker -> trackers_sent
shared give1_aID: trackers_sent -> interact [Coercer1_aID_tracker=1]
shared give2_aID: trackers_sent -> interact [Coercer1_aID_tracker=2]
shared not_give_aID: trackers_sent -> interact [Coercer1_aID_tracker=0]
shared punish_aID: interact -> ckeck [aID_punish=true]
shared not_punish_aID: interact -> check [aID_punish=false]
shared check_tracker1_aID: check -> end
shared check_tracker2_aID: check -> end
PROTOCOL: [[coerce1_aID, coerce2_aID, coerce3_aID], [punish, not_punish]]
```

Formula

$$\varphi_{vuln,i,k} = \langle\langle Coercer \rangle\rangle \mathrm{G}((end \wedge revote_{v1} = k \wedge voted_{v1} = i) \rightarrow \mathrm{K}_{Coercer} voted_{vi} = i)$$

Configurations:

- First candidate ($i = 1$) and $k = \#R$ revotes
- Last candidate ($i = \#C$) and $k = \#R$ revotes
- First candidate ($i = 1$) and $k = \#R - 1$ revotes
- Last candidate ($i = \#C$) and $k = \#R - 1$ revotes

#A	#R	Full Model					Reduced Model					Result
		#st	#tr	Seq.	Par.	Appr.	#st	#tr	Seq.	Par.	Appr.	
4	3	3.63e4	7.46e4	0.003	0.009	1.121	2.60e4	5.99e4	0.001	0.002	0.184	True
4	5	5.62e4	1.15e5	0.004	0.003	0.345	4.01e4	9.26e4	0.002	0.002	0.283	True
4	10	1.06e5	2.18e5	0.009	0.005	0.691	7.55e4	1.74e5	0.004	0.002	0.563	True
5	3	1.55e6	5.91e6	0.158	0.004	14.78	1.09e6	4.65e6	0.112	0.021	12.99	True
6	3	7.61e7	4.98e8	0.524	0.051	41.24	5.34e7	3.82e8	0.427	0.042	37.35	True
7	3	model generation timeout					model generation timeout					-

Table 1

Verification of $\varphi_{vuln,i,k}$ for the first candidate ($i = 1$) and $k = \#R$ revotes

#Ag	#R	Full Model					Reduced Model					Result
		#st	#tr	Seq.	Par.	Appr.	#st	#tr	Seq.	Par.	Appr.	
4	3	3.63e4	7.46e4	0.003	0.010	1.103	2.60e4	5.99e4	0.002	0.003	0.166	True
4	5	5.62e4	1.15e5	0.004	0.005	0.348	4.01e4	9.26e4	0.003	0.003	0.280	True
4	10	1.06e5	2.18e5	0.008	0.009	0.700	7.55e4	1.74e5	0.005	0.004	0.567	True
5	3	1.55e6	5.91e6	0.160	0.055	14.03	1.09e6	4.65e6	0.112	0.053	12.49	True
6	3	7.61e7	4.98e8	0.602	0.083	42.44	5.34e7	3.82e8	0.501	0.057	38.20	True
7	3	model generation timeout					model generation timeout					-

Table 2

Verification of $\varphi_{vuln,i,k}$ for the last candidate ($i = \#C$) and $k = \#R$ revotes

#Ag	#R	Full Model					Reduced Model					Result
		#st	#tr	Seq.	Par.	Appr.	#st	#tr	Seq.	Par.	Appr.	
4	3	3.63e4	7.46e4	0.303	0.317	1.128	2.60e4	5.99e4	0.202	0.205	0.179	False
4	5	5.62e4	1.15e5	0.524	0.592	0.325	4.01e4	9.26e4	0.411	0.503	0.280	False
4	10	1.06e5	2.18e5	0.721	0.668	0.459	7.55e4	1.74e5	0.525	0.512	0.364	False
5	3	1.55e6	5.91e6	2.146	1.257	0.981	1.09e6	4.65e6	1.513	1.003	0.583	False
6	3	7.61e7	4.98e8	5.232	3.228	1.892	5.34e7	3.82e8	4.986	2.427	1.092	False
7	3	model generation timeout					model generation timeout					-

Table 3
Verification of $\varphi_{vuln,i,k}$ for the first candidate ($i = 1$) and $k = \#R - 1$ revotes

#Ag	#R	Full Model					Reduced Model					Result
		#st	#tr	Seq.	Par.	Appr.	#st	#tr	Seq.	Par.	Appr.	
4	3	3.63e4	7.46e4	0.302	0.311	0.180	2.60e4	5.99e4	0.201	0.213	0.126	False
4	5	5.62e4	1.15e5	0.519	0.584	0.310	4.01e4	9.26e4	0.410	0.475	0.283	False
4	10	1.06e5	2.18e5	0.742	0.627	0.462	7.55e4	1.74e5	0.558	0.544	0.370	False
5	3	1.55e6	5.91e6	2.160	1.358	0.942	1.09e6	4.65e6	1.621	1.009	0.519	False
6	3	7.61e7	4.98e8	5.504	3.516	1.903	5.34e7	3.82e8	5.110	2.380	1.112	False
7	3	model generation timeout					model generation timeout					-

Table 4

Verification of $\varphi_{vuln,i,k}$ for the last candidate ($i = \#C$) and $k = \#R - 1$ revotes

Results

- DominoDFS and alternative distributed algorithm performed much slower and are omitted from the results
- Parallel verification performs quite well in most cases
- Performance of the parallel algorithm depends heavily on the structure of the model
- The fixpoint approximation performs well in cases where no strategy can be found

Conclusions

Modal logics for MAS are characterized by high computational complexity.

We used the „all out” approach, verifying a genuine protocol for secure voting.

Partial-order reductions, simple DFS, simple distributed DFS and fixpoint approximation show very promising performance.

Thank you for your attention!