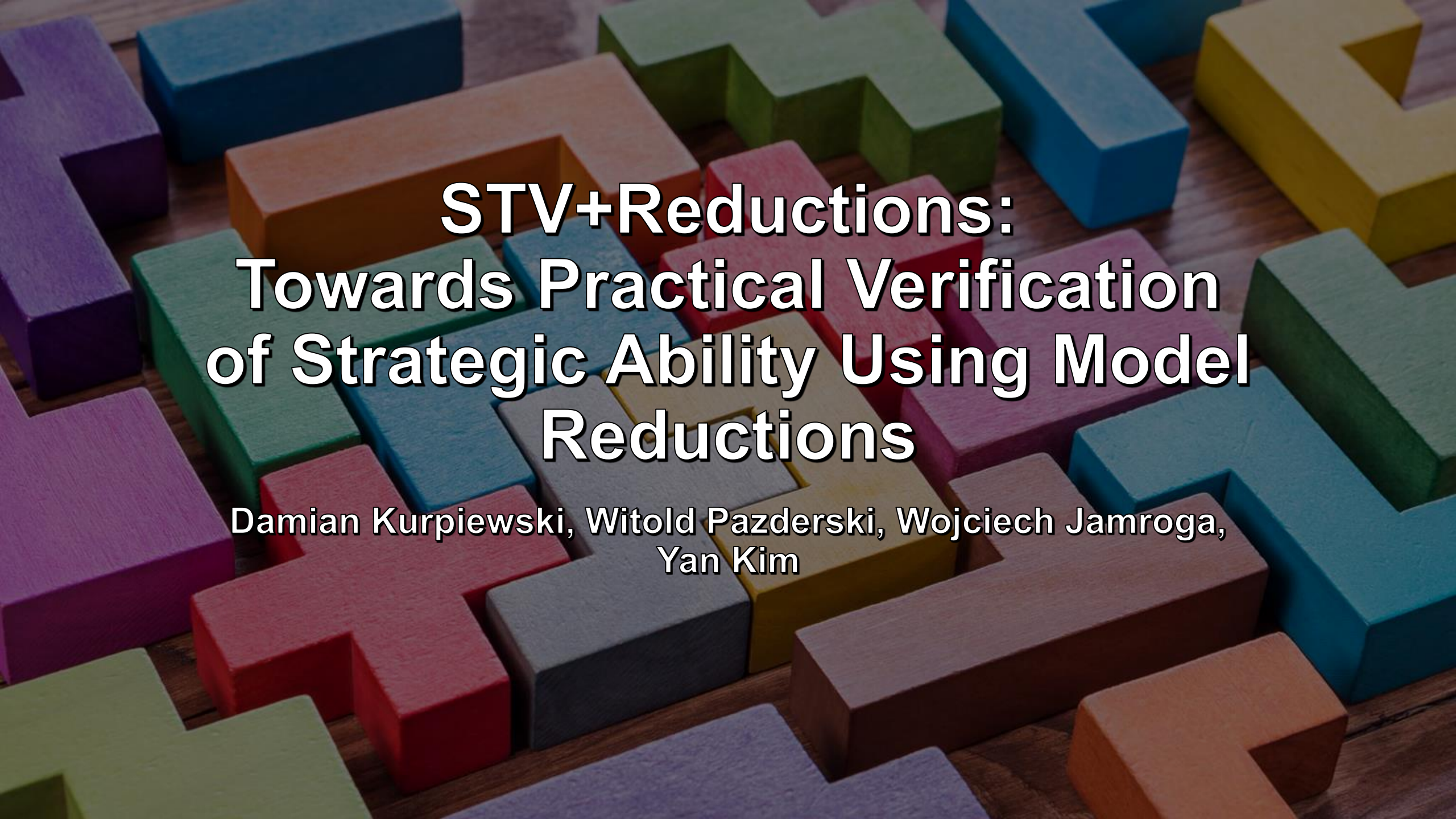


STV+Reductions: Towards Practical Verification of Strategic Ability Using Model Reductions

Damian Kurpiewski, Witold Pazderski, Wojciech Jamroga, Yan Kim

STV: What's that?

- StraTegic Verifier
- Model-checker, mostly for $\mathrm{ATL}_{\mathrm{ir}}$
- Pre-alpha version
- Main (command line) part is not user friendly
- ☺ Has a nice **graphical interface**
- Open source

Previous Version

- Presented at AAMAS 2019 (Demo session)
- Included several pre-configured scenarios (TianJi, Castles, Bridge Endplay, Drones, Simple Voting)
- User could:
  - Generate the model with custom parameters and view it
  - Use apprixomations or DominoDFS for verification

Current Version

- New methods:
  - Partial order reductions
  - Bisimulation checking
- Simple specification language
- Asynchronous models
- Interface changes

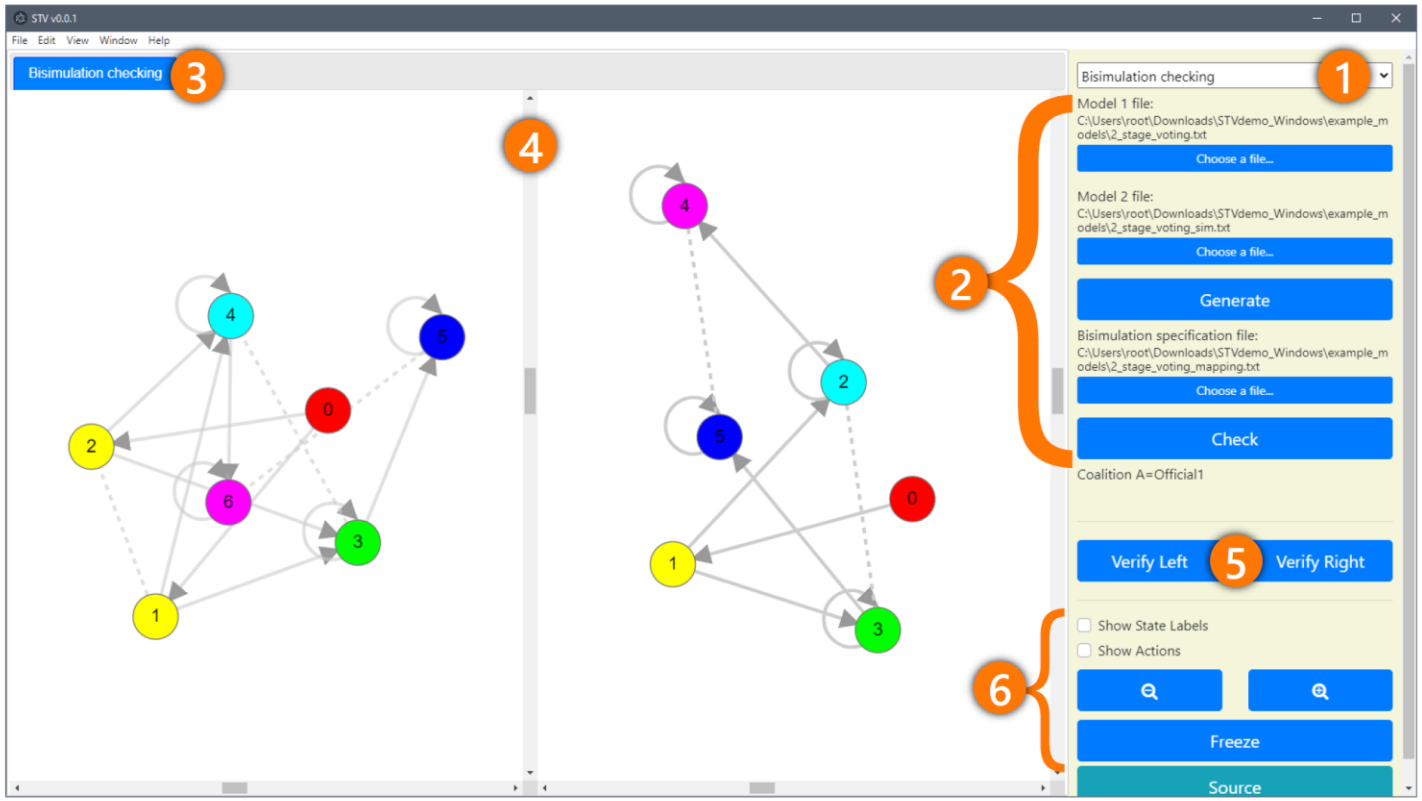
STV: Model Checking for Strategies under Imperfect Information (Demo)

Wojciech Jamroga, Damian Kurpiewski, Witold Pazderski, Yan Kim

{w.jamroga, d.kurpiewski, w.pazderski}@ipipan.waw.pl, yan.kim@uni.lu

Institute of Computer Science
Polish Academy of Sciences

University of Luxembourg



1 CHOOSE AN ACTION

STV is a tool for verification of Multi-Agent Systems. It does explicit-state model checking and addresses the state space explostion problem. STV offers:

- model verification,
- automated partial order reduction,
- bisimulation - checking equivalence of models according to a defined relation of A-bisimulation.

2 GENERATE MODEL(S)

STV includes several parameterized example models:

- asynchronous: simple & two-stage voting, train-gate-controller;
- synchronous: bridge end-play, castles, drones, Tian Ji.

In a model specification file user can define:

- local automata for the agent(s),
- propositional variables,
- persistent propositions,
- agent names,
- ATL formula.

3 CHOOSE A MODEL TO VIEW:

- local automaton for each agent,
- generated global and reduced models for POR,
- side-by-side view of two (global) models for bisimuation-checking.

4 EXPLORE MODEL(S)

GUI provides an intuitive interface with color-highlights for:

- initial states,
- winning strategy (if exists),
- states satisfying given formula,
- reduced model fragment,
- pairs of bisimilar node subsets.

5 VERIFY MODEL(S)

Given formula can be verified both on global and reduced models using:

- fix-point approximation (upper/lower),
- dominance-based strategy search (DominoDFS).

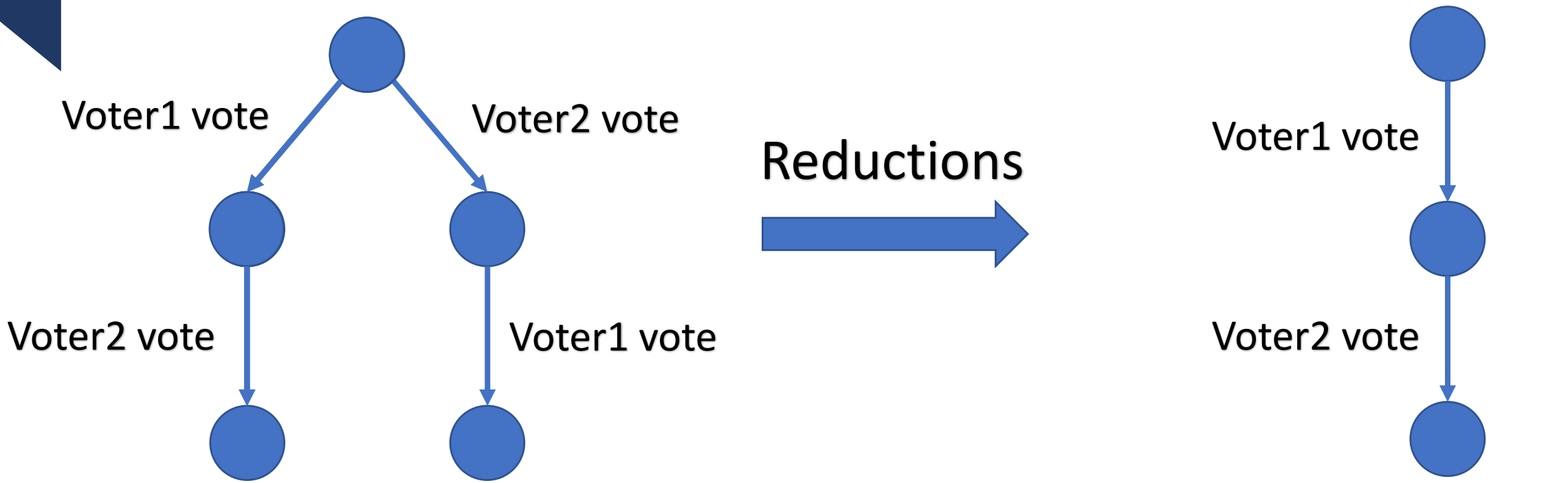
6 ADJUST GRAPH SETTINGS

- The view can be panned and zoomed.
- Labels with state or transition details can be shown by hovering over the target node/edge or toggled for the whole graph.

Partial Order Reductions

- **Fully automated**
- Greatly **reduces the state-space** for some models
- Model is reduced according to the given formula
- Works on asynchronous models

Example: Voting



Bisimulation checking

- „**Expert mode**” reductions

Bisimulation checking

- „**Expert mode**” reductions
- **Fully automated bisimulation checking**

Bisimulation checking

- „**Expert mode**” reductions

- **Fully automated bisimulation checking**
- **…but the reduced model must be created by hand**

Bisimulation checking

- „**Expert mode**” reductions

- **Fully automated bisimulation checking**
- **…but the reduced model must be created by hand**

- User provides:
  - Initial model
  - Reduced model
  - States mapping
  - Coalition

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Shared transitions

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Shared transitions

Local transitions

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Shared transitions

Local transitions

States (templates)

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Shared transitions

Local transitions

States (templates)

Proposition variables

Simple Specification Language

```
Agent Train[2]:
init: wait
shared a1_aID: wait -> tunnel [aID_in=true]
shared a2_aID: tunnel -> away [aID_in=false]
a3: away -> wait

Agent Controller[1]:
init: green
shared a1_Train1: green -> red
shared a1_Train2: green -> red
shared a2_Train1: red -> green
shared a2_Train2: red -> green

REDUCTION: [in_Train1,in_Train2]
COALITION: [Controller1]
FORMULA: <<Controller1>>F(Train1_in=True | Train2_in=True)
```

Agents (templates)

Initial states

Shared transitions

Local transitions

States (templates)

Proposition variables

Configuration