Specification and Verification of MAS for Local Communities

Michael Köster and Peter Lohmann

Computational Intelligence Group,
Clausthal University of Technology
ICR Seminar, December 13, 2011

Outline

Motivation

IT Ecosystems

- **Classical approaches** do not scale well for today's **large and complex software-intensive systems**.
- Software systems are **connected** among each other and **interact** massively.

⇝ **IT Ecosystem**:

- analog to **biological ecosystems**
- based on the **balance between individuals** (autonomy) and **sets of rules** (control) defining **equilibria** within an **IT Ecosystem**
- **Maintaining** and continuously evolving IT Ecosystems requires deep understanding of this balance.

IT Ecosystems

- **Classical approaches** do not scale well for today's **large and complex software-intensive systems**.
- Software systems are **connected** among each other and **interact** massively.

⇝ **IT Ecosystem**:

- analog to **biological ecosystems**
- based on the **balance between individuals** (autonomy) and **sets of rules** (control) defining **equilibria** within an **IT Ecosystem**
- **Maintaining** and continuously evolving IT Ecosystems requires deep understanding of this balance.

Application

In **Smart Cities** are the following **IT Ecosystems**:

- Smart-Living-Systems
- Smart-Working-Systems
- Smart-Transport-Systems
- Smart-Energy-Systems
- etc.

IT Ecosystems Project

NTH focused Research School for IT Ecosystems:

- Technische Universität Braunschweig,
- Technische Universität Clausthal,
- Leibniz Universität Hannover.

Three main projects:

- **AIM**: Buttom-Up Approach
  Adaptive Information methods.
- **RuleIT**: Top-Down Approach
  Rules are infered from the design phase and verified at runtime.
- **Loccom**: Combination of both: Buttom-Up and Top-Down approaches.

Local Communities

Nowadays **Social Networks**:

- exchange of information,
- groups of interests, and
- explicit use of a computer or smartphone.

Local Communities: social networks + real social networks

- exchange of information works automatically,
- spontaneous and dynamic groups of interests, and
- implicit use of a smart phone.

Local Communities

Nowadays **Social Networks**:

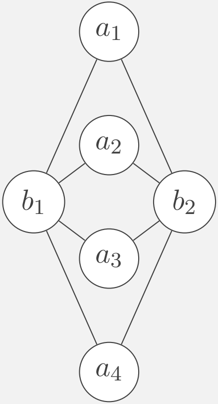
- exchange of information,
- groups of interests, and
- explicit use of a computer or smartphone.

Local Communities: **social networks** + **real social networks**

- exchange of information works automatically,
- spontaneous and dynamic groups of interests, and
- implicit use of a smart phone.

Communication between Agents

Example 1 (Communicating Agents)

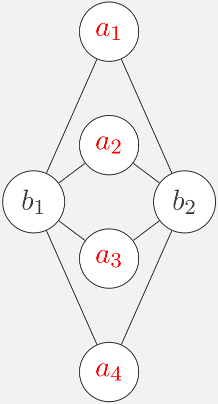


- Agents of team B are not allowed to send a message back.
- If an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round, $k > i$.

Is it possible for team A to ensure that a_4 will know the message eventually?

Communication between Agents

Example 1 (Communicating Agents)

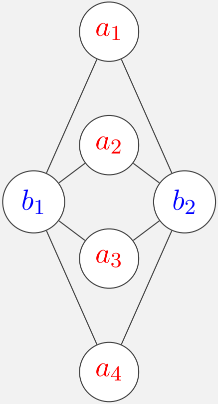


- Agents of team B are not allowed to send a message back.
- If an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round, $k > i$.

Is it possible for team A to ensure that a_4 will know the message eventually?

Communication between Agents

Example 1 (Communicating Agents)

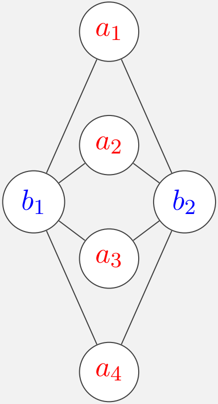


- Agents of team B are not allowed to send a message back.
- If an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round, $k > i$.

Is it possible for team A to ensure that a_4 will know the message eventually?

Communication between Agents

Example 1 (Communicating Agents)

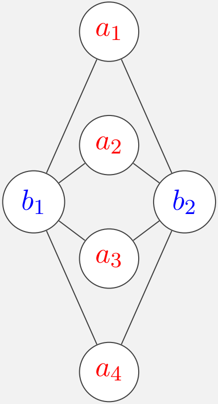


- Agents of team B are not allowed to send a message back.
- If an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round, $k > i$.

Is it possible for team A to ensure that a_4 will know the message eventually?

Communication between Agents

Example 1 (Communicating Agents)



- Agents of team B are not allowed to send a message back.
- If an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round, $k > i$.

Is it possible for team A to ensure that a_4 will know the message eventually?

Issues

How can we

- formalize such a system?
- deal with big systems?
- formulate minimal properties (fairness, liveness, safety)?
- ensure that the minimal properties hold?

Issues

How can we

- formalize such a system?
  Modular Interpreted System (MIS)
- deal with big systems?
- formulate minimal properties (fairness, liveness, safety)?
- ensure that the minimal properties hold?

Issues

How can we

- formalize such a system?
  Modular Interpreted System (MIS)
- deal with big systems?
  Abstraction for MIS
- formulate minimal properties (fairness, liveness, safety)?
- ensure that the minimal properties hold?

Issues

How can we

- formalize such a system?
  Modular Interpreted System (MIS)
- deal with big systems?
  Abstraction for MIS
- formulate minimal properties (fairness, liveness, safety)?
  Alternating-Time Temporal Logic (ATL)
- ensure that the minimal properties hold?

Issues

How can we

- formalize such a system?
  Modular Interpreted System (MIS)
- deal with big systems?
  Abstraction for MIS
- formulate minimal properties (fairness, liveness, safety)?
  Alternating-Time Temporal Logic (ATL)
- ensure that the minimal properties hold?
  Model Checking

Model Checking

Advantages:

- Faster than **Theorem Proving**

Disadvantages:

- Explicit representation: a lot of states, in practise not usable
- Compact representation: model checking complexity increases

Idea:

- Reduce the state space before model checking

Modular Interpreted System

Modular Interpreted System

Idea:

- Several **loosely** connected components.
- Work is done in the components.
- Little **interaction** between the components.

Modular Interpreted System

Definition 2 (Modular Interpreted System (MIS))

A MIS is a tuple $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$ where:

- $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ agents,
- Act actions,
- $\mathcal{I}n$ interaction alphabet.

Modular Interpreted System

Definition 2 (Modular Interpreted System (MIS))

A MIS is a tuple $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$ where:

- $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ agents,
- Act actions,
- $\mathcal{I}n$ interaction alphabet.

Example 3

- $\mathbb{A}\mathrm{gt} = \{a_1, a_2, a_3, a_4, b_1, b_2\}$,
- $Act = \{\mathrm{send_x} \mid \mathrm{x} \in \mathbb{A}\mathrm{gt}\} \cup \{\mathrm{noop}\}$
- $\mathcal{I}n = \{\mathbf{nothing}, \mathbf{m_{a_1}}, \mathbf{m_{a_2}}, \mathbf{m_{a_3}}, \mathbf{m_{a_4}}\} \cup \mathrm{P}(\{\mathbf{m_{a_i b_j}} \mid i \in \{1, \ldots, 4\}, j \in \{1, 2\}\})$

MIS Agent I

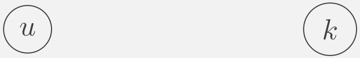
Definition 4 (MIS Agent)

Each agent has the following **internal structure**:

$$a_i = (St_i, \quad , \quad , \quad , \quad , \quad)$$

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 5 (Agent a_1)



MIS Agent I

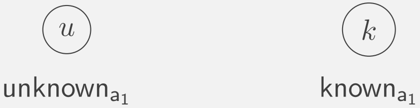
Definition 4 (MIS Agent)

Each agent has the following **internal structure**:

$$a_i = (St_i, \quad , \quad\quad , \quad\quad , \quad , \Pi_i, \pi_i)$$

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 5 (Agent a_1)



MIS Agent I

Definition 4 (MIS Agent)

Each agent has the following **internal structure**:

$$a_i = (St_i, d_i, \quad , \quad , \quad , \Pi_i, \pi_i)$$

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 5 (Agent a_1)

MIS Agent I

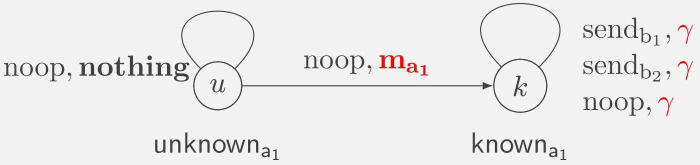
Definition 4 (MIS Agent)

Each agent has the following **internal structure**:

$$a_i = (St_i, d_i, \quad , in_i, o_i, \Pi_i, \pi_i)$$

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 5 (Agent a_1)



MIS Agent I

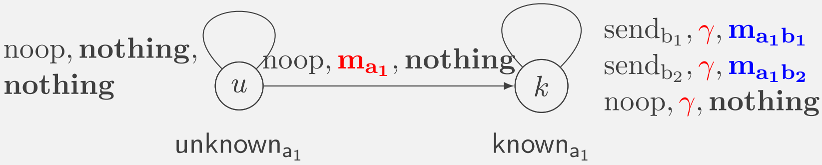
Definition 4 (MIS Agent)

Each agent has the following **internal structure**:

$$a_i = (St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i)$$

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 5 (Agent a_1)



MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{\mathsf{b_j}}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with known_{b_j} if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{b_j}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with known_{b_j} if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{b_j}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{\mathsf{b_j}}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{\mathsf{b_j}}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

Example 6 (Agents b_j)

- 256 states
- Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{\mathsf{b_j}}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

MIS Agent II

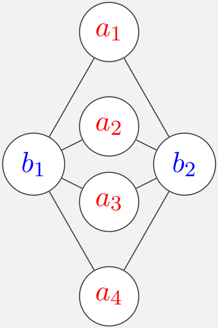
Example 6 (Agents b_j)

- 256 states
- Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the agent received some time ago the message from a_i.
- Others are labeled with $\mathsf{unknown}_{\mathsf{b_j}}$.
- While the agent is waiting for a message it does nothing.
- When it receives a message it has to send the message to one of the opponents with a higher number then i.
- If the agent received the message from each agent a_i it does nothing.

Advantages of Modular Interpreted Systems I

- **Modular** (removing, replacing of agents)
- **Interaction** reduced to an abstract **interaction symbol**
- **Computational ground**

Example 7 (Communicating Agents)

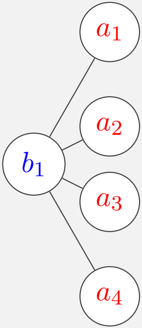


Is it possible for team A to ensure that a_4 will know the message eventually?

Advantages of Modular Interpreted Systems I

- **Modular** (removing, replacing of agents)
- **Interaction** reduced to an abstract **interaction symbol**
- **Computational ground**

Example 7 (Communicating Agents)



Is it possible for team A to ensure that a_4 will know the message eventually?

Advantages of Modular Interpreted Systems II

Example 8 (Communicating Agents)

- Simple example
- $256 + 256 + 2 + 2 + 2 + 2 = 520$ states
- Remove agent b_2: 264 states

Specification Logic: ATL

Alternating-time Temporal Logic

Definition 9 (Alternating-time temporal Logic (ATL))

The language of plain ATL is defined over the non-empty sets:

- Π of **Propositions** $\mathsf{p} \in \Pi$
- $\mathbb{A}\mathrm{gt}$ of **Agents** $A \subseteq \mathbb{A}\mathrm{gt}$

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle \mathbf{X}\varphi \mid \langle\langle A\rangle\rangle \mathbf{G}\varphi \mid \langle\langle A\rangle\rangle \varphi \mathbf{U}\varphi.$$

Main Idea: cooperation modalities

- $\langle\langle A\rangle\rangle \mathbf{X}\varphi$ "coalition A has a collective strategy to enforce that φ holds after the next step"

Alternating-time Temporal Logic

Definition 9 (Alternating-time temporal Logic (ATL))

The language of plain ATL is defined over the non-empty sets:

- Π of **Propositions** $\mathsf{p} \in \Pi$
- $\mathbb{A}\mathrm{gt}$ of **Agents** $A \subseteq \mathbb{A}\mathrm{gt}$

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle \mathbf{X}\varphi \mid \langle\langle A\rangle\rangle \mathbf{G}\varphi \mid \langle\langle A\rangle\rangle \varphi \mathbf{U}\varphi.$$

Main Idea: **cooperation modalities**

- $\langle\langle A\rangle\rangle \mathbf{X}\varphi$ "coalition A has a collective strategy to enforce that φ holds after the next step"

Example 10

Is it possible for team A to ensure that a_4 will know the message eventually?

$$S, \mathrm{q} \models \langle\langle A \rangle\rangle(\top \mathbf{U} \mathsf{known}_{\mathsf{a}_4})$$

Abstraction for MIS

Basic Idea I

Partitioning the local state space by using handcrafted equivalence relation:

- Reducing all equivalent states to just one
- Agents get fewer choices and the opponents more choices
- Influence symbols of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- Local transition function:
  Input: abstract state, action, influence symbol
  Output: abstract state
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:
  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:
  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:
  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:
  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:

  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea I

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Reducing all equivalent states to just one
- Agents get **fewer choices** and the **opponents more choices**
- **Influence symbols** of the abstract states:
  - all influence symbols that are an outcome of executing an action in each concrete state
- **Local transition function**:

  Input: **abstract state**, **action**, **influence symbol**
  Output: **abstract state**
  - unfold both equivalence classes
  - connection between concrete state of 1st equivalence class to concrete state of 2nd equivalence class?

Basic Idea II

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Labeling:
  - Assigning a proposition to an abstract state if all concrete states in the equivalence class are labeled with that proposition
  - If a proposition only holds in some states of the class we remove it from set of propositions

Basic Idea II

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Labeling:
  - Assigning a proposition to an abstract state if all concrete states in the equivalence class are labeled with that proposition
  - If a proposition only holds in some states of the class we remove it from set of propositions

Basic Idea II

Partitioning the **local state space** by using **handcrafted equivalence relation**:

- Labeling:
  - Assigning a proposition to an abstract state if all concrete states in the equivalence class are labeled with that proposition
  - If a proposition only holds in some states of the class we remove it from set of propositions

Abstraction Relation

Definition 11 (Abstraction Relation)

An **abstraction relation** for a MIS is a product $\equiv = \equiv_1 \times \cdots \times \equiv_k$ where each $\equiv_i \subseteq St_i \times St_i$ is an equivalence relation for the states St_i of agent a_i.

For $q \in St_i$, we write $[q]_{\equiv_i}$ for the equivalence class of the local state q with respect to $\equiv_i$. And for $q \in St = St_1 \times \cdots \times St_k$, we write $[q]_{\equiv}$ for the equivalence class of the global state q.

Definition 12 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_{\equiv} := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $St'_i := \{[q]_{\equiv_i} \mid q \in St_i\}$
- $d'_i([q]_{\equiv_i}) := \begin{cases} \bigcap_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \in A \\ \bigcup_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \notin A \end{cases}$
- $out'_i([q]_{\equiv_i}, \alpha) := \bigcup_{q' \in [q]_{\equiv_i}} out_i(q', \alpha)$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 12 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_\equiv := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $St'_i \;:=\; \{[q]_{\equiv_i} \mid q \in St_i\}$
- $d'_i([q]_{\equiv_i}) \;:=\; \begin{cases} \bigcap_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \in A \\ \bigcup_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \notin A \end{cases}$
- $out'_i([q]_{\equiv_i}, \alpha) \;:=\; \bigcup_{q' \in [q]_{\equiv_i}} out_i(q', \alpha)$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 12 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_\equiv := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $St'_i \;:=\; \{[q]_{\equiv_i} \mid q \in St_i\}$
- $d'_i([q]_{\equiv_i}) \;:=\; \begin{cases} \bigcap_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \in A \\ \bigcup_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \notin A \end{cases}$
- $out'_i([q]_{\equiv_i}, \alpha) \;:=\; \bigcup_{q' \in [q]_{\equiv_i}} out_i(q', \alpha)$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 13 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_\equiv := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $in'_i([q]_{\equiv_i}, \gamma_1, \ldots, \gamma_{k-1}) := \bigcup_{q' \in [q]_{\equiv_i}} in_i(q', \gamma_1, \ldots, \gamma_{k-1})$
- $o'_i([q]_{\equiv_i}, \alpha, \gamma) := \bigcup_{q' \in [q]_{\equiv_i}} \{[q'']_{\equiv_i} \mid q'' \in o_i(q', \alpha, \gamma)\}$
- $\Pi'_i := \Pi_i \cap \{p_i \mid \forall q \in \pi_i(p_i) : \forall q' \in [q]_{\equiv_i} : q' \in \pi_i(p_i)\}$
- $\pi'_i(p_i) := \{[q]_{\equiv_i} \mid q \in \pi_i(p_i)\}$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 13 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\text{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\text{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_\equiv := (\mathbb{A}\text{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\text{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $in'_i([q]_{\equiv_i}, \gamma_1, \ldots, \gamma_{k-1}) \; := \bigcup_{q' \in [q]_{\equiv_i}} in_i(q', \gamma_1, \ldots, \gamma_{k-1})$
- $o'_i([q]_{\equiv_i}, \alpha, \gamma) \; := \bigcup_{q' \in [q]_{\equiv_i}} \{[q'']_{\equiv_i} \mid q'' \in o_i(q', \alpha, \gamma)\}$
- $\Pi'_i \; := \; \Pi_i \cap \{p_i \mid \forall q \in \pi_i(p_i) : \forall q' \in [q]_{\equiv_i} : \; q' \in \pi_i(p_i)\}$
- $\pi'_i(p_i) \; := \; \{[q]_{\equiv_i} \mid q \in \pi_i(p_i)\}$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 13 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_{\equiv} := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \dots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $in'_i([q]_{\equiv_i}, \gamma_1, \dots, \gamma_{k-1}) := \bigcup_{q' \in [q]_{\equiv_i}} in_i(q', \gamma_1, \dots, \gamma_{k-1})$
- $o'_i([q]_{\equiv_i}, \alpha, \gamma) := \bigcup_{q' \in [q]_{\equiv_i}} \{[q'']_{\equiv_i} \mid q'' \in o_i(q', \alpha, \gamma)\}$
- $\Pi'_i := \Pi_i \cap \{p_i \mid \forall q \in \pi_i(p_i) : \forall q' \in [q]_{\equiv_i} : q' \in \pi_i(p_i)\}$
- $\pi'_i(p_i) := \{[q]_{\equiv_i} \mid q \in \pi_i(p_i)\}$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \dots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Definition 13 (Abstraction for MIS)

For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of **favored** agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the **abstraction** of S with respect to $\equiv$ and A as the MIS

$$S^A_{\equiv} := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

- $in'_i([q]_{\equiv_i}, \gamma_1, \ldots, \gamma_{k-1}) := \bigcup_{q' \in [q]_{\equiv_i}} in_i(q', \gamma_1, \ldots, \gamma_{k-1})$
- $o'_i([q]_{\equiv_i}, \alpha, \gamma) := \bigcup_{q' \in [q]_{\equiv_i}} \{[q'']_{\equiv_i} \mid q'' \in o_i(q', \alpha, \gamma)\}$
- $\Pi'_i := \Pi_i \cap \{p_i \mid \forall q \in \pi_i(p_i) : \forall q' \in [q]_{\equiv_i} : q' \in \pi_i(p_i)\}$
- $\pi'_i(p_i) := \{[q]_{\equiv_i} \mid q \in \pi_i(p_i)\}$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Example for Abstraction

Example 14 (Agents b_j)

- 256 states

$$S_i := \{(R,N) \mid \emptyset \neq R \subseteq \{r_1,\dots,r_i\}, n_i \in N\} \setminus \bigcup_{j=1}^{i-1} S_j \quad \text{for } i = 1,\dots,3$$

$$S_{\text{rest}} := \{(R,N) \mid (R,N) \notin S_1 \cup \dots \cup S_3\}$$

Result: **4 states**

Agent b_1

Example 15 (Communicating Agents I)

$$b'_1 = (St'_{b_1}, d'_{b_1}, out'_{b_1}, in_{b_1}, o'_{b_1}, \Pi_{b_1}, \pi'_{b_1})$$

where

- $St'_{b_1} \quad = \quad \{S_1, S_2, S_3, S_{\text{rest}}\}$
- $\pi'_{b_1} : \quad S_i \mapsto \{\mathsf{known}_{\mathsf{b}_1}\}$ for $i = 1, \ldots, 3$
  $S_{\text{rest}} \mapsto \{\mathsf{known}_{\mathsf{b}_1}, \mathsf{unknown}_{\mathsf{b}_1}\}$
- $d'_{b_1}(S_i) = \{\mathrm{send_x} \mid \mathrm{x} \in \{a_{i+1}, \ldots, a_4\}\}$ for $i = 1, \ldots, 3$
  $d'_{b_1}(S_{\text{rest}}) = \{\mathrm{send_x} \mid \mathrm{x} \in \{a_1, \ldots, a_4\}\} \cup \{\mathrm{noop}\}$

Agent b_1

Example 15 (Communicating Agents I)

$$b'_1 = (St'_{b_1}, d'_{b_1}, out'_{b_1}, in_{b_1}, o'_{b_1}, \Pi_{b_1}, \pi'_{b_1})$$

where

- $St'_{b_1} \quad = \quad \{S_1, S_2, S_3, S_{\text{rest}}\}$
- $\pi'_{b_1} : \quad S_i \mapsto \{\mathsf{known}_{\mathsf{b_1}}\}$ for $i = 1, \ldots, 3$
  $S_{\text{rest}} \mapsto \{\mathsf{known}_{\mathsf{b_1}}, \mathsf{unknown}_{\mathsf{b_1}}\}$
- $d'_{b_1}(S_i) \quad = \quad \{\text{send}_\text{x} \mid \text{x} \in \{a_{i+1}, \ldots, a_4\}\}$ for $i = 1, \ldots, 3$
  $d'_{b_1}(S_{\text{rest}}) \quad = \quad \{\text{send}_\text{x} \mid \text{x} \in \{a_1, \ldots, a_4\}\} \cup \{\text{noop}\}$

Agent b_1

Example 15 (Communicating Agents I)

$$b_1' = (St'_{b_1}, d'_{b_1}, out'_{b_1}, in_{b_1}, o'_{b_1}, \Pi_{b_1}, \pi'_{b_1})$$

where

- $St'_{b_1} \;=\; \{S_1, S_2, S_3, S_{\text{rest}}\}$
- $\pi'_{b_1}: \; S_i \mapsto \{\mathsf{known}_{\mathsf{b_1}}\}$ for $i = 1, \ldots, 3$
  $S_{\text{rest}} \mapsto \{\mathsf{known}_{\mathsf{b_1}}, \mathsf{unknown}_{\mathsf{b_1}}\}$
- $d'_{b_1}(S_i) \;=\; \{\text{send}_\text{x} \mid \text{x} \in \{a_{i+1}, \ldots, a_4\}\}$ for $i = 1, \ldots, 3$
  $d'_{b_1}(S_{\text{rest}}) \;=\; \{\text{send}_\text{x} \mid \text{x} \in \{a_1, \ldots, a_4\}\} \cup \{\text{noop}\}$

Agent b_1

Example 16 (Communicating Agents II)

- $$\begin{aligned} out'_{b_1} : \quad (S_{\text{rest}}, \text{noop}) &\mapsto \{\mathbf{nothing}\} \\ (s, \text{send}_{a_1}) &\mapsto \{\mathbf{m_{b_1 a_1}}\} \\ (s, \text{send}_{a_2}) &\mapsto \{\mathbf{m_{b_1 a_2}}\} \\ (s, \text{send}_{a_3}) &\mapsto \{\mathbf{m_{b_1 a_3}}\} \\ (s, \text{send}_{a_4}) &\mapsto \{\mathbf{m_{b_1 a_4}}\} \end{aligned}$$

  for all $s \in St'_{b_1}$,

- $$\begin{aligned} o'_{b_1} : \quad (S_{\text{rest}}, \alpha, \mathbf{nothing}) &\mapsto \{S_{\text{rest}}\} \\ (S_{\text{rest}}, \alpha, \{\mathbf{m_{a_j b_1}}\}) &\mapsto \{S_{\text{rest}}, S_j\} \\ (S_i, \text{send}_{a_j}, \mathbf{nothing}) &\mapsto \{S_{\text{rest}}\} \\ (S_i, \text{send}_{a_j}, \{\mathbf{m_{a_{j'} b_1}}\}) &\mapsto \{S_{j'}\} \\ (S_i, \text{send}_{a_j}, \{\mathbf{m_{a_4 b_1}}\}) &\mapsto \{S_{\text{rest}}\} \end{aligned}$$

  for all $\alpha \in Act$, $\gamma \in \mathcal{I}n$, $i, j' \in \{1,2,3\}$ and $j \in \{1,2,3,4\}$.

Agent b_1

Example 16 (Communicating Agents II)

- $$\begin{array}{rcll} out'_{b_1} : & (S_{\text{rest}}, \text{noop}) & \mapsto & \{\mathbf{nothing}\} \\ & (s, \text{send}_{\text{a}_1}) & \mapsto & \{\mathbf{m_{b_1 a_1}}\} \\ & (s, \text{send}_{\text{a}_2}) & \mapsto & \{\mathbf{m_{b_1 a_2}}\} \\ & (s, \text{send}_{\text{a}_3}) & \mapsto & \{\mathbf{m_{b_1 a_3}}\} \\ & (s, \text{send}_{\text{a}_4}) & \mapsto & \{\mathbf{m_{b_1 a_4}}\} \end{array}$$
  for all $s \in St'_{b_1}$,
- $$\begin{array}{rcll} o'_{b_1} : & (S_{\text{rest}}, \alpha, \mathbf{nothing}) & \mapsto & \{S_{\text{rest}}\} \\ & (S_{\text{rest}}, \alpha, \{\mathbf{m_{a_j b_1}}\}) & \mapsto & \{S_{\text{rest}}, S_j\} \\ & (S_i, \text{send}_{\text{a}_\text{j}}, \mathbf{nothing}) & \mapsto & \{S_{\text{rest}}\} \\ & (S_i, \text{send}_{\text{a}_\text{j}}, \{\mathbf{m_{a_{j'} b_1}}\}) & \mapsto & \{S_{j'}\} \\ & (S_i, \text{send}_{\text{a}_\text{j}}, \{\mathbf{m_{a_4 b_1}}\}) & \mapsto & \{S_{\text{rest}}\} \end{array}$$
  for all $\alpha \in Act$, $\gamma \in \mathcal{I}n$, $i, j' \in \{1, 2, 3\}$ and $j \in \{1, 2, 3, 4\}$.

Advantages of Abstraction of Modular Interpreted Systems

Example 17 (Communicating Agents III)

- $256 + 256 + 2 + 2 + 2 + 2 = 520$ states
- Remove agent b_2: 264 states
- Abstraction of b_1: 12 states

The Model Checking Algorithm

TU Clausthal
Claustal University of Technology

Idea

- **Input**:
  - MIS S
  - $init$ of global states of S
  - ATL formula φ
  - for each quantifier subformulae an abstraction relation
- **Output**:
  - true : if $S, q \models \varphi$ for all $q \in init$
  - unknown : we do not know whether S satisfies φ or not

Idea

- **Input**:
  - MIS S
  - $init$ of global states of S
  - ATL formula φ
  - for each quantifier subformulae an abstraction relation
- **Output**:
  - true : if $S, q \models \varphi$ for all $q \in init$
  - unknown : we do not know whether S satisfies φ or not

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta_i'$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := label(\theta_i, \equiv_{\theta_i})$
  - Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \dots, \theta_n, \ell_1, \dots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \dots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta_i'$, and
- $\ell_1, \dots, \ell_m$ are literals.
- For all $i \in \{1, \dots, n\}$ do:

  Set $W_i := label(\theta_i, \equiv_{\theta_i})$

  Set $S := S(w_i, W_i)$; a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \dots, w_n, \ell_1, \dots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B\rangle\rangle\theta_i'$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  Set $W_i := label(\theta_i, \equiv_{\theta_i})$.
  Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta'_i$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  Set $W_i := label(\theta_i, \equiv_{\theta_i})$
  Set $S := S(w_i, W_i)$; a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta_i'$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i :=$ *label*$(\theta_i, \equiv_{\theta_i})$.
  - Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta_i'$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i :=$ *label*$(\theta_i, \equiv_{\theta_i})$.
  - Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta'_i$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i :=$ *label*$(\theta_i, \equiv_{\theta_i})$.
  - Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: modelcheck

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ Boolean formula (conjunctions and disjunctions only),
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier, i.e. each θ_i is of the form $\langle\langle B \rangle\rangle \theta_i'$, and
- $\ell_1, \ldots, \ell_m$ are literals.
- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i :=$ *label*$(\theta_i, \equiv_{\theta_i})$.
  - Set $S := S(w_i, W_i)$, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
- If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in init$ then return true. Otherwise return unknown.

Algorithm: label

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y} \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

Algorithm: label

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y} \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

Algorithm: label

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y} \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

Algorithm: label

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y} \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

Algorithm: label

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y} \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \dots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^\psi \langle\langle A\rangle\rangle \mathbf{Y} \lambda(w_1, \dots, w_n, \ell_1, \dots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^\psi \langle\langle A \rangle\rangle \mathbf{Y} \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^\psi \langle\langle A \rangle\rangle \mathbf{Y} \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^{\psi} \langle\langle A \rangle\rangle \mathbf{Y} \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\text{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^\psi \langle\langle A \rangle\rangle \mathbf{Y} \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Algorithm: label II

- Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

- For all $i \in \{1, \ldots, n\}$ do:
  - Set $W_i := \{[q]_\equiv \mid \forall q' \in [q]_\equiv : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.
  - Set $S' := S'(w_i, W_i)$.
- Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_\equiv \mid S', [q]_\equiv \models \neg^\psi \langle\langle A\rangle\rangle \mathbf{Y} \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

  by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm.
- Return $W := \{q \in St \mid [q]_\equiv \in W'\}$.

Complexity and Soundness

Theorem 18

Algorithm modelcheck$(S, init, \varphi, (\equiv_\psi)_{\psi\in\mathrm{qsf}(\varphi)})$ *runs in time*

$$O\left(|init| + |S| \cdot |\varphi|\right) \cdot 2^{O\left(\sum_{\psi\in\mathrm{qsf}(\varphi)} \left|S^{[\![\psi]\!]}_{\equiv_\psi}\right|\right)}$$

where $|S|$ *denotes the size of the* MIS S *in a compact representation. The cardinality of the global state space of* S *may then be upto* $2^{\Theta(|S|)}$.

Theorem 19

Algorithm modelcheck is sound, i.e. if modelcheck$(S, init, \varphi,$ $(\equiv_\psi)_{\psi\in\mathrm{qsf}(\varphi)})$ *outputs* true *then* $S, q \models \varphi$ *for all* $q \in init$.

Conclusion

Conclusion

- **IT Ecosystems** are inspired by **biological ecosystems**.
- **Modular Interpreted Systems** facilitates **modularity**
- **ATL** allows to talk about strategic properties
- The **abstraction for MIS** :
  - is sound
  - allows to deal with bigger Systems
    (depending on the equivalent relations)

Thank you for your attention!

Questions?

Conclusion

- **IT Ecosystems** are inspired by **biological ecosystems**.
- **Modular Interpreted Systems** facilitates **modularity**
- **ATL** allows to talk about strategic properties
- The **abstraction for MIS** :
  - is sound
  - allows to deal with bigger Systems
    (depending on the equivalent relations)

Thank you for your attention!

Questions?

Conclusion

- **IT Ecosystems** are inspired by **biological ecosystems**.
- **Modular Interpreted Systems** facilitates **modularity**
- **ATL** allows to talk about strategic properties
- The **abstraction for MIS** :
  - is sound
  - allows to deal with bigger Systems
    (depending on the equivalent relations)

Thank you for your attention!

Questions?

References

W. Jamroga, T. Agotnes
Modular Interpreted Systems
Proceedings of AAMAS'07, pp. 892-899.

Rajeev Alur and Thomas A. Henzinger and Orna Kupferman
Alternating-time temporal logic
J. ACM, Volume 49, Number 5, 2002, pp. 672-713.

Michael Köster, Peter Lohmann
Abstraction for Model Checking Modular Interpreted Systems over ATL (Extended Abstract)
Proceedings of AAMAS'11

Michael Köster, Peter Lohmann
Abstraction for Model Checking Modular Interpreted Systems over ATL
Proceedings of Programming Multi-Agent Systems, AAMAS'11