Algorithmic Verification for Epistemic Logic

Ron van der Meyden
University of New South Wales/National ICT Australia
Algorithmic Verification for Epistemic Logic
Informatics North American Summer School in Logic Language and

1

Slide 1

Examples and MCK System Demonstration
Lecture 4

Slide 2

n agents

Slide 3

\( n \) agents

\( \text{ACT}_1 \times \ldots \times \text{ACT}_n \)

Joint actions: \( \text{ACT} \times \ldots \times \text{ACT}_n \)

\( a \) set of actions for each agent \( i \in [1 \ldots n] \)

Slide 4

A finite interpreted environment for \( n \) agents is a tuple of the form

\[ E \equiv \langle S_e, I_e, \tau, O, \pi_e, \alpha \rangle \]

where the components are as follows:

1. \( S_e \) is a finite set of states of the environment.
2. \( I_e \) is a subset of \( S_e \) the initial states of the environment.
3. \( \tau \) is a function mapping joint actions \( a \in \text{ACT} \) to state transition relations \( \tau(a) \subseteq S_e \times S_e \).
4. \( O \equiv \langle O_1, \ldots, O_n \rangle \) is a tuple of observation functions \( O_i: S_e \rightarrow \text{Obs} \).
5. \( \pi_e: S_e \times \text{Prop} \rightarrow \{0, 1\} \) is an interpretation.
6. \( \alpha \subseteq S_e \) is a Büchi acceptance condition.
A protocol for agent $i$ is a function $P_i: S_i \to P_i(R_i)$. A joint protocol $P$ is a tuple $\langle P_1, \ldots, P_n \rangle$, where each $P_i$ is a protocol for agent $i$.

A run of a joint protocol $P$ in an environment $E$ is an infinite sequence $\varepsilon = s_0 s_1 \ldots$ of states of $E$ such that:
1. $s_0 \in I_e$,
2. For all $k \geq 0$, there exists a joint action $a = \langle a_1, \ldots, a_n \rangle$ such that $(s_k, s_{k+1}) \in \tau(a)$ and $a_i \in P_i(r_{[0\ldots k]})$,
3. Some $\sigma \in \mathcal{S}$ occurs infinitely often.

In all cases $\varepsilon_v(m) = \varepsilon(m)$

Examples:
1. The observational view: $\varepsilon_{\text{obs}}_i(m) = O_i(\varepsilon(m))$
2. The clock view: $\varepsilon_{\text{obs}}_i(m) = (m, O_i(\varepsilon(m)))$
3. The synchronous perfect recall view: $\varepsilon_{\text{spr}}_i(m) = O_i(\varepsilon(0)) \ldots O_i(\varepsilon(m))$

Let $e$ be an event of $d$, a view associates a local state with each agent at each point of time, determining a mapping $e: \mathbb{N} \to L_n \times S_e$.

System generated by an environment with a view

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{A}f$</td>
<td>$\mathcal{A}$ in all next states.</td>
</tr>
<tr>
<td>$\mathcal{E}f$</td>
<td>$\mathcal{E}$ in at least one next state.</td>
</tr>
<tr>
<td>$\mathcal{A}[f U g]$</td>
<td>$\mathcal{A}$ on all paths, $f$ until $g$.</td>
</tr>
<tr>
<td>$\mathcal{E}[f U g]$</td>
<td>$\mathcal{E}$ on at least one path, $f$ until $g$.</td>
</tr>
<tr>
<td>$\mathcal{A}f$</td>
<td>$\mathcal{A}$ in all next states.</td>
</tr>
<tr>
<td>$\mathcal{E}f$</td>
<td>$\mathcal{E}$ in all next states.</td>
</tr>
</tbody>
</table>

3. The synchronous perfect recall view:
$$\varepsilon_{\text{spr}}(m) = (\varepsilon(m), 0)$$

2. The clock view:
$$\varepsilon_{\text{clock}}(m) = (m, \varepsilon(m))$$

1. The observational view:
$$\varepsilon_{\text{obs}}(m) = \varepsilon(m)$$
- Food moves under control of the environment at most one step per unit time.
- Sensors ∈ \{position, position+1\}

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>eventually ( f ).</td>
</tr>
<tr>
<td>( g )</td>
<td>always ( f ).</td>
</tr>
<tr>
<td>( U )</td>
<td>until ( g ).</td>
</tr>
<tr>
<td>( X )</td>
<td>in the next state.</td>
</tr>
<tr>
<td>( f ) in ( n ) steps.</td>
<td></td>
</tr>
</tbody>
</table>

**Language**
- **Observational**
  - Specdock
  - Spec: Perfect Recall
  - TE: Timing
  - U: Unlabeled
  - LTL: Linear Temporal Logic

**Clock**
- Specdock

**Sync. Recall**
- Specdock

**Perfect Recall**
- Specdock

**Perfect Recall**
- Specdock

**Perfect Recall**
- Specdock

A knowledge-based program:

\[ \text{wait until } \text{Know(position in Goal); halt.} \]

Implementations when \( \text{Goal} = \{2, 3, 4\} \) and agent's view = \( \text{Sensor} \):

- I1: wait until Sensor = 3; halt.
- I2: wait until Sensor in \{3, 4, 5\}; halt.

**Dining Cryptographers**

David Chaum, J. Cryptology 1988:

Three cryptographers are sitting down to dinner at their favorite three-star restaurant. The waiter informs them that arrangements have been made for the bill to be paid anonymously. One of the cryptographers might be paying for the entire meal, or it might have been the NSA (US National Security Agency). The three cryptographers respect each other’s rights to pay anonymously, but they wonder if the NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol:

1. Each cryptographer flips a coin behind his menu, between him and the cryptographer to his right, so that only the two of them can see the outcome. Each cryptographer then states aloud whether the two coins fell on the same side or different sides.
2. If one of the cryptographers is the payer, he states the opposite of what he sees.
3. An odd number of differences uttered at the table indicates that the NSA is paying, an even number of differences indicates that at most one cryptographer is paying.

If a cryptographer is paying neither of the other two learns anything new from the utterances about which cryptographer it is.

**Protocol:**

They resolve their uncertainty fairly by carrying out the following protocol. They wonder if the NSA is paying.

Dining cryptographers respect each other's right to anonymously pay for the entire meal, or it might have been the NSA (US National Security Agency). The cryptographers are at their favorite three-star restaurant. The waiter informs them that arrangements have been made for the bill to be paid anonymously. One of the cryptographers might be paying for the entire meal, or it might have been the NSA (US National Security Agency). The three cryptographers respect each other’s right to pay anonymously, but they wonder if the NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol:

1. Each cryptographer flips a coin behind his menu, between him and the cryptographer to his right, so that only the two of them can see the outcome.
2. Each cryptographer then states aloud whether the two coins fell on the same side or different sides.
3. If one of the cryptographers is the payer, he states the opposite of what he sees.
4. An odd number of differences uttered at the table indicates that the NSA is paying, an even number of differences indicates that at most one cryptographer is paying.

If a cryptographer is paying neither of the other two learns anything new from the utterances about which cryptographer it is.

**Protocol:**

They resolve their uncertainty fairly by carrying out the following protocol. They wonder if the NSA is paying.

Dining cryptographers respect each other's right to anonymously pay for the entire meal, or it might have been the NSA (US National Security Agency). The cryptographers are at their favorite three-star restaurant. The waiter informs them that arrangements have been made for the bill to be paid anonymously. One of the cryptographers might be paying for the entire meal, or it might have been the NSA (US National Security Agency). The three cryptographers respect each other’s right to pay anonymously, but they wonder if the NSA is paying. They resolve their uncertainty fairly by carrying out the following protocol:

1. Each cryptographer flips a coin behind his menu, between him and the cryptographer to his right, so that only the two of them can see the outcome.
2. Each cryptographer then states aloud whether the two coins fell on the same side or different sides.
3. If one of the cryptographers is the payer, he states the opposite of what he sees.
4. An odd number of differences uttered at the table indicates that the NSA is paying, an even number of differences indicates that at most one cryptographer is paying.

If a cryptographer is paying neither of the other two learns anything new from the utterances about which cryptographer it is.