Undecidability in Epistemic Planning

Guillaume Aucher, IRISA, University of Rennes 1 – INRIA, France
Thomas Bolander, DTU Compute, Tech Univ of Denmark
Introduction

Our paper in a nutshell:

What we have shown: Undecidability of planning when allowing (arbitrary levels of) higher-order reasoning (epistemic planning). Higher-order reasoning here means reasoning about the beliefs of yourself and other agents (and nesting of such).

How we have shown it: Reduction of the halting problem for two-counter machines.

Structure of talk:

1. Motivation.
2. Introducing the basics: planning + logic + two-counter machines.
3. Sketching the proof: How to encode two-counter machines as epistemic planning problems.
4. Summary of results.
Planning and higher-order reasoning

Automated planning: Given a goal formula, an initial state and some actions, compute a sequence of actions that leads from the initial state to a state satisfying the goal formula.

Example.
Goal: On(A,B) ∧ On(B,C).

![Diagram showing the planning process with initial state C B A, goal A B C, and actions Put(c,table), Put(b,table), Put(b,c), and Put(a,b).]
Planning and higher-order reasoning

Automated planning: Given a **goal formula**, an **initial state** and some **actions**, compute a sequence of actions that leads from the initial state to a state satisfying the goal formula.

**Example.**
**Goal:** On(A,B) ∧ On(B,C).

Why higher-order reasoning in planning?
**Our framework for planning with higher-order reasoning**

**Epistemic planning**: Our framework for planning with higher-order reasoning.

From **classical planning** to **epistemic planning**: Replace the propositional logic underlying classical planning by **Dynamic Epistemic Logic (DEL)**.

<table>
<thead>
<tr>
<th></th>
<th>Classical planning</th>
<th>Epistemic planning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>States</strong></td>
<td>models of prop. logic</td>
<td>models of MA epist. logic</td>
</tr>
<tr>
<td><strong>Goal formula</strong></td>
<td>formula of prop. logic</td>
<td>formula of MA epist. logic</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td>induced by action schemas</td>
<td>event models of DEL</td>
</tr>
</tbody>
</table>

Epistemic planning can deal with: non-determinism, partial observability, sensing actions, multiple agents, higher-order reasoning.
DEL by example: A private announcement

epistemic model

**DEL by example: A private announcement**

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
**DEL by example: A private announcement**

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
- **Product update**: As in [Baltag et al., 1998].
Event models: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.

Event model above: Private announcement of \( p \) to agent 0.

Product update: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.

- **Event model above**: Private announcement of $p$ to agent 0.

- **Product update**: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
- **Product update**: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of \( p \) to agent 0.
- **Product update**: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
- **Product update**: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
- **Product update**: As in [Baltag et al., 1998].
DEL by example: A private announcement

- **Event models**: Only preconditions, no postconditions. Means: Purely epistemic planning, no change of ontic facts.
- **Event model above**: Private announcement of $p$ to agent 0.
- **Product update**: As in [Baltag et al., 1998].
- **In resulting model**: Agent 0 knows $p$ ($\square_0 p$ holds), but agent 1 didn’t learn anything.
Planning interpretation of DEL

- **Epistemic states**: Pointed, finite epistemic models.
Planning interpretation of DEL

- **Epistemic states**: Pointed, finite epistemic models.
- **Epistemic actions**: Pointed, finite event models.
Planning interpretation of DEL

- **Epistemic states**: Pointed, finite epistemic models.
- **Epistemic actions**: Pointed, finite event models.
- **Result of applying an action in a state**: Product update of state with action.
Epistemic planning tasks and plan existence problem

Definition
An epistemic planning task is \((s_0, A, \phi_g)\), where

- \(s_0\) is the initial state: an epistemic state.
- \(A\) is a finite set of epistemic actions.
- \(\phi_g\) is the goal formula: a formula of epistemic logic.
Definition
An epistemic planning task is \((s_0, A, \phi_g)\), where

- \(s_0\) is the initial state: an epistemic state.
- \(A\) is a finite set of epistemic actions.
- \(\phi_g\) is the goal formula: a formula of epistemic logic.

Definition
A solution to a planning task \((s_0, A, \phi_g)\) is a sequence of actions \(a_1, \ldots, a_n \in A\) such that \(s_0 \otimes a_1 \otimes \cdots \otimes a_n \models \phi_g\).
An epistemic planning task is \((s_0, A, \phi_g)\), where

- \(s_0\) is the initial state: an epistemic state.
- \(A\) is a finite set of epistemic actions.
- \(\phi_g\) is the goal formula: a formula of epistemic logic.

A solution to a planning task \((s_0, A, \phi_g)\) is a sequence of actions \(a_1, \ldots, a_n \in A\) such that \(s_0 \otimes a_1 \otimes \cdots \otimes a_n \models \phi_g\).

The plan existence problem in epistemic planning is the following decision problem “Given an epistemic planning task \((s_0, A, \phi_g)\), does it have a solution?”

We will now show undecidability of the plan existence problem...
Two-counter machines

Configurations: \[ k \mid l \mid m \], where \( k, l, m \in \mathbb{N} \).

Instruction set: \( \text{inc}(0), \text{inc}(1), \text{jump}(j), \text{jzdec}(0,j), \text{jzdec}(1,j), \text{halt} \).

Computation step example:

\[
\begin{array}{c|c|c}
  k & l & m \\
\end{array} \quad \text{inc}(0) \quad \begin{array}{c|c|c}
  k+1 & l+1 & m \\
\end{array}
\]

The halting problem for two-counter machines is undecidable [Minsky, 1967].
Proof idea for undecidability of epistemic planning

Our proof idea is this. For each two-register machine, construct a corresponding planning task where:

- The **initial state** encodes the initial configuration of the machine.
- The **actions** encode the instructions of the machine.
- The **goal formula** is true of all epistemic states representing halting configurations of the machine.

Then show that the two-register machine halts iff the corresponding planning task has a solution. (Execution paths of the planning task encodes computations of the machine).
Encodings

Encoding configurations as epistemic states:

\[
\begin{align*}
&k \mid l \mid m \\
\hookrightarrow & \begin{cases}
  k + 1 \text{ worlds} & \{ p_1, p_1, p_1, p_1 \} \\
  l + 1 \text{ worlds} & \{ p_2, p_2, p_2 \} \\
  m + 1 \text{ worlds} & \{ p_3, p_3, p_3 \}
\end{cases}
\end{align*}
\]
Encodings

Encoding configurations as epistemic states:

\[
\begin{array}{c|c|c}
  k & l & m \\
\end{array}
\sim
\begin{cases}
  k + 1 \\
  l + 1 \\
  m + 1 \\
\end{cases}
\]

Encoding instructions as epistemic actions:

\[ \neg (p_1 \lor p_2 \lor p_3) \]

\[ \text{inc}(0) \sim \begin{cases}
  p_1 \land \Diamond \top \\
  p_1 \land \Diamond \Box \bot \\
  p_1 \land \Box \bot \\
\end{cases} \]

\[ \begin{cases}
  p_2 \land \Diamond \top \\
  p_2 \land \Diamond \Box \bot \\
  p_2 \land \Box \bot \\
\end{cases} \]

\[ p_3 \]
The computation step \( \begin{array}{c} k \\ l \\ m \end{array} \rightarrow \begin{array}{c} k + 1 \\ l + 1 \\ m \end{array} \) is mimicked by:

\[
\text{encoding}(\begin{array}{c} k \\ l \\ m \end{array}) \otimes \text{encoding}(\text{inc(0)}) = \]

\[
\neg(p_1 \lor p_2 \lor p_3)
\]

\[
\begin{array}{c}
p_1 \\ p_2 \\ p_3
\end{array}
\]

\[
\begin{array}{c}
p_1 \\ p_2 \\ p_3
\end{array}
\]

\[
\begin{array}{c}
p_1 \\ p_2 \\ p_3
\end{array}
\]
The computation step $k_l_m$ is mimicked by:

$$encoding(k_l_m) \otimes encoding(\text{inc}(0)) =$$

$$\neg(p_1 \lor p_2 \lor p_3)$$
The computation step

\[
\begin{array}{c|c|c}
  k & l & m \\
  \downarrow & \downarrow & \downarrow \\
  k+1 & l+1 & m
\end{array}
\]

is mimicked by:

\[
\text{encoding}(\begin{array}{c|c|c}
  k & l & m \\
\end{array}) \otimes \text{encoding}(\text{inc}(0)) =
\]

\[
\neg (p_1 \lor p_2 \lor p_3)
\]

\[
\begin{array}{c|c|c}
  \ldots & \ldots & \ldots \\
  \downarrow & \downarrow & \downarrow \\
  \ldots & \ldots & \ldots
\end{array}
\]

\[
\begin{array}{c|c|c}
  p_1 & p_2 & p_3 \\
  \downarrow & \downarrow & \downarrow \\
  p_1 & p_2 & p_3
\end{array}
\]

\[
\begin{array}{c|c|c}
  \ldots & \ldots & \ldots \\
  \downarrow & \downarrow & \downarrow \\
  \ldots & \ldots & \ldots
\end{array}
\]

\[
\begin{array}{c|c|c}
  p_1 & p_2 & p_3 \\
  \downarrow & \downarrow & \downarrow \\
  p_1 & p_2 & p_3
\end{array}
\]

\[
\begin{array}{c|c|c}
  \ldots & \ldots & \ldots \\
  \downarrow & \downarrow & \downarrow \\
  \ldots & \ldots & \ldots
\end{array}
\]

\[
\begin{array}{c|c|c}
  p_1 & p_2 & p_3 \\
  \downarrow & \downarrow & \downarrow \\
  p_1 & p_2 & p_3
\end{array}
\]

\[
\begin{array}{c|c|c}
  \ldots & \ldots & \ldots \\
  \downarrow & \downarrow & \downarrow \\
  \ldots & \ldots & \ldots
\end{array}
\]

\[
\begin{array}{c|c|c}
  p_1 & p_2 & p_3 \\
  \downarrow & \downarrow & \downarrow \\
  p_1 & p_2 & p_3
\end{array}
\]
The computation step $k l m \xrightarrow{\text{inc}(0)} k + 1 l + 1 m$ is mimicked by:

$$encoding([k \ l \ m]) \otimes encoding(\text{inc}(0)) =$$

\[
\begin{align*}
\neg(p_1 \lor p_2 \lor p_3) \otimes \\
p_1 \land \Diamond \top & \quad p_2 \land \Diamond \top \quad p_3 \\
p_1 \land \Box \bot & \quad p_2 \land \Box \bot \\
p_1 & \quad p_2 & \quad p_3
\end{align*}
\]
The computation step
\[ \begin{array}{c|c|c} k & l & m \\ \hline \end{array} \xrightarrow{\text{inc}(0)} \begin{array}{c|c|c} k + 1 & l + 1 & m \\ \hline \end{array} \]
is mimicked by:

\[
\text{encoding}(\begin{array}{c|c|c} k & l & m \end{array}) \otimes \text{encoding}(\text{inc}(0)) = \]

\[
\neg (p_1 \lor p_2 \lor p_3) \]

\[
\begin{array}{c|c|c} k + 1 \end{array} \xrightarrow{p_1} \begin{array}{c|c} l + 1 \end{array} \xrightarrow{p_2} \begin{array}{c} m + 1 \end{array} \xrightarrow{p_3} \]

\[
\begin{array}{c|c|c} k + 1 \end{array} \xrightarrow{p_1} \begin{array}{c} l + 1 \end{array} \xrightarrow{p_2} \begin{array}{c} m + 1 \end{array} \xrightarrow{p_3} \]

The computation step $\begin{bmatrix} k & l & m \end{bmatrix}$ is mimicked by:

$\text{encoding}(\begin{bmatrix} k & l & m \end{bmatrix}) \otimes \text{encoding}(\text{inc}(0)) = \neg(p_1 \lor p_2 \lor p_3)$
The computation step $k \mid l \mid m$ mimicked by:

$\text{encoding}(k \mid l \mid m) \otimes \text{encoding}(\text{inc}(0)) =$
The computation step \( k \mid l \mid m \rightarrow k + 1 \mid l + 1 \mid m \) is mimicked by:

\[
\text{encoding}(k \mid l \mid m) \otimes \text{encoding}(\text{inc}(0)) =
\]

\[
\neg (p_1 \lor p_2 \lor p_3) \otimes (p_1 \land \lozenge \top \land p_2 \land \lozenge \top \land p_3 \land \lozenge \false) =
\]

\[
\neg (p_1 \lor p_2 \lor p_3) \otimes (p_1 \land \lozenge \false \land p_2 \land \lozenge \false) =
\]

\[
\text{encoding}(k + 1 \mid l + 1 \mid m)
\]
Summary of results on (un)decidability of plan existence in epistemic planning

<table>
<thead>
<tr>
<th>L</th>
<th>transitive</th>
<th>Euclidean</th>
<th>reflexive</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>KT</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>K4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>K45</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>S4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>S5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Theorem

The figure to the right summarises our results on decidability (D) and undecidability (UD) of the plan existence problem in epistemic planning.

<table>
<thead>
<tr>
<th></th>
<th>Single-agent planning</th>
<th>Multi-agent planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>UD</td>
<td>UD</td>
</tr>
<tr>
<td>KT</td>
<td>UD</td>
<td>UD</td>
</tr>
<tr>
<td>K4</td>
<td>UD</td>
<td>UD</td>
</tr>
<tr>
<td>K45</td>
<td>D</td>
<td>UD</td>
</tr>
<tr>
<td>S4</td>
<td>UD</td>
<td>UD</td>
</tr>
<tr>
<td>S5</td>
<td>D</td>
<td>UD</td>
</tr>
</tbody>
</table>

Summary

• We prove that **allowing arbitrary levels of higher-order reasoning leads to undecidability of planning**. Even in the propositional and purely epistemic case.

• Essence of the problem: Even if your beliefs are strengthened through your actions, it might just mean that you ignorance is pushed to deeper and deeper levels. And we can put no bound on this depth of ignorance, hence no bound on depth (size) of epistemic states.
**Corollary: Undecidability of model checking in $\mathcal{L}_{\text{DEL}}^*$**

The *DEL* language $\mathcal{L}_{\text{DEL}}^*$ is defined by the following BNF:

\[
\begin{align*}
\phi & ::= p \mid \neg \phi \mid (\phi \land \phi) \mid \Box_i \phi \mid [\pi] \phi \\
\pi & ::= (E, e) \mid (\pi \cup \pi) \mid (\pi; \pi) \mid \pi^*
\end{align*}
\]

where $p \in P$, $i \in A$ and $(E, e)$ is any epistemic action [van Ditmarsch et al., 2007].

**Theorem**

The model checking problem of the language $\mathcal{L}_{\text{DEL}}^*$ is undecidable.

**Proof.**

The plan existence problem considered above is reducible to the model checking problem of $\mathcal{L}_{\text{DEL}}^*$. \qed