Epistemic reasoning in AI

François Schwarzentruber

École Normale Supérieure Rennes

IJCAI-ECAI, Tutorial T27, 14 July 2018
Talks at IJCAI-ECAI 2018

- Game Description Language and Dynamic Epistemic Logic Compared. Thorsten Engesser, Robert Mattmüller, Bernhard Nebel, Michael Thielscher
- Single-Shot Epistemic Logic Program Solving. Manuel Bichler, Michael Morak, Stefan Woltran
- Model Checking Probabilistic Epistemic Logic for Probabilistic Multiagent Systems. Chen Fu, Andrea Turrini, Xiaowei Huang, Lei Song, Yuan Feng, Lijun Zhang
- The Complexity of Limited Belief Reasoning—The Quantifier-Free Case Yijia Chen, Abdallah Saffidine, Christoph Schwering
- Small Undecidable Problems in Epistemic Planning Sébastien Lê Cong, Sophie Pinchinat, _
- Multi-agent Epistemic Planning with Common Knowledge Qiang Liu, Yongmei Liu
Objective of this tutorial

1. Being able to understand these IJCAI-ECAI papers in the field
2. Being able to model epistemic multi-agent scenarios
3. Being able to contribute in the field
4. Promote automatic structures for proving decidability
   [Blumensath and Grädel 2000]
5. (if time) Advertise knowledge-base programs for writing policies
Many different settings

This tutorial is not a catalogue (although this slide is one):

- QdecPOMDP, decPOMDP [Brafman, Shani, and Zilberstein 2013]
- Belief revision [Alchourrón, Gärdenfors, and Makinson 1985]
- ATL with imperfect information [Hoek and Wooldridge 2003]
- Epistemic situation calculus [Scherl and Levesque 2003]
- Game Description Logic III [Thielscher 2016]
- Dynamic epistemic logic [Baltag, Moss, and Solecki 1998]
- Probabilistic Dynamic epistemic logic [B. P. Kooi 2003]
- Interpreted systems [Fagin et al. 1995]
- Explicit and implicit beliefs [Lorini 2018]

Why we focus on Dynamic epistemic logic?

1. Action-oriented: it extends classical planning;
2. Has a nice classification of different decision problems.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)
- Epistemic states
- Epistemic languages
- Actions
- Update product
- Dynamic language
- Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)
- Epistemic states
- Epistemic languages
- Actions
- Update product
- Dynamic language
- Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Examples of epistemic states

http://hintikkasworld.irisa.fr/

[demo IJCAI-ECAI 2018]
Epistemic states

[van Ditmarsch, van der Hoek, and B. Kooi 2008]

Let \( AP = \{ p, p_1, \ldots \} \) be a countable set of atomic propositions.
Let \( AGT = \{ a, b, c, \ldots \} \) be a finite set of agents.

Definition

An epistemic model \( \mathcal{M} = (W, (R_a)_{a \in AGT}, V) \) is a tuple where:

- \( W = \{ w, u, \ldots \} \) is a non-empty set of possible worlds;
- \( R_a \subseteq W \times W \) is an accessibility relation for agent \( a \);
- \( V : W \rightarrow 2^{AP} \) is a valuation function.

A pair \( (\mathcal{M}, w) \) is called a epistemic state, where \( w \) represents the actual world.
Example

Modeling using Dynamic Epistemic Logic (DEL)

Epistemic states
Epistemic languages
Actions
Update product
Dynamic language
Succinct models
Bounded epistemic planning
Unbounded epistemic planning
Automatic structures for decidability of unbounded epistemic planning when propositional pre/post
Knowledge-based programs
Conclusion
References

\[ ▶ W = \{ w, u, v, s \}; \]
\[ ▶ R_a = \{ (w, w), (w, u), (u, w), (u, u), (v, v), (v, s), (s, v), (s, s) \}; \]
\[ ▶ R_b = \{ (w, w), (w, v), (v, w), (v, v), (u, u), (u, s), (s, u), (s, s) \}; \]
\[ ▶ V(w) = \{ dirty_a, dirty_b \}; \]
\[ ▶ V(u) = \{ dirty_b \}; \]
\[ ▶ V(v) = \{ dirty_a \}; \]
\[ ▶ V(s) = \emptyset. \]
Outline

Modeling using Dynamic Epistemic Logic (DEL)
  Epistemic states
  Epistemic languages
  Actions
  Update product
  Dynamic language
  Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Syntax of $\mathcal{L}_{EL}$

**Definition**

The syntax of $\mathcal{L}_{EL}$ is given by the following grammar:

$$
\varphi, \psi, \ldots ::= p \mid \neg \varphi \mid (\varphi \lor \psi) \mid K_a \varphi
$$

where $p$ ranges over $AP$ and $a$ ranges over $AGT$.

The size of $\varphi$ is the number of symbols needed to write $\varphi$.

**Notation (Dual operators)**

- $(\varphi \land \psi)$ for $\neg(\neg \varphi \lor \neg \psi)$;
- $\hat{K}_a \varphi$ for $\neg K_a \neg \varphi$.

- $K_a \varphi$ is read ‘agent $a$ knows/believes that $\varphi$ is true;
- $\hat{K}_a \varphi$ is read ‘agent $a$ considers $\varphi$ as possible’.

**Definition**

$L_{Prop}$ is the set of propositional logic formulas.
Semantics of $\mathcal{L}_{EL}$

Definition

The semantics of $\mathcal{L}_{EL}$ is defined as follows:

- $\mathcal{M}, w \models p$ if $p \in V(w)$;
- $\mathcal{M}, w \models \neg \varphi$ if it is not the case that $\mathcal{M}, w \models \varphi$;
- $\mathcal{M}, w \models (\varphi \lor \psi)$ if $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$;
- $\mathcal{M}, w \models K_a \varphi$ if for all $u$ s.t. $w R_a u$, $\mathcal{M}, u \models \varphi$.

$\mathcal{M}, w \models K_a \text{dirty}_b$
Dual operators

\[ \mathcal{M}, w \models K_a \varphi \quad \text{if for all u s.t. } wR_u, \mathcal{M}, u \models \varphi \]

\[ \mathcal{M}, w \models \hat{K}_a \varphi \quad \text{if there exists u s.t. } wR_u, \mathcal{M} \text{ and } u \models \varphi. \]

\[ \mathcal{M}, w \models K_a \text{dirty}_b \]

\[ \mathcal{M}, w \models \hat{K}_a \text{dirty}_a \]
Common knowledge

Common knowledge of $\varphi$ among agents in group $G$

Definition
The syntax of $\mathcal{L}_{\text{ELCK}}$ is given by the following grammar:

$$\varphi ::= p \mid \neg \varphi \mid (\varphi \lor \varphi) \mid K_a \varphi \mid C_G \varphi$$

where $p$ ranges over $AP$, $a$ ranges over $AGT$, and $G$ ranges over $2^{AGT}$.

Definition
The semantics of $\mathcal{L}_{\text{ELCK}}$ extended by the following clause:

- $\mathcal{M}, w \models C_G \varphi$ if for all $u \in W$, $wR_G u$ implies $\mathcal{M}, u \models \varphi$

where $R_G$ is the transitive closure of $\bigcup_{a \in G} R_a$. 
Outline

Modeling using Dynamic Epistemic Logic (DEL)
- Epistemic states
- Epistemic languages

Actions
- Update product
- Dynamic language
- Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Examples of actions

Example (Public announcement of \("p\)"")

![Diagram of a public announcement action]

Example (Private announcement \("p\) to \(a\)"")

![Diagram of a private announcement action]

[Baltag, Moss, and Solecki 1998]

Example (Public announcement of \("p\)"")

- **Pre:** \(p\)
- **Post:** \(-\)

Example (Private announcement \("p\) to \(a\)"")

- **Pre:** \(p\)
- **Post:** \(-\)
- **Pre:** \(true\)
- **Post:** \(-\)
Examples of actions

Example (Transfer marble from basket to box)

pre : inBasket
post : inBasket := false
pre : inBox
post : inBox := true

\[ a \]
\[ b \]

pre : true
post : \text{false}

\[ a, b \]
**Actions**

![Diagram showing actions with pre: p, post: − → b → pre: true, post: − → a, b]

**Definition**

An event model $\mathcal{E} = (E, (R^E_a)_{a \in AGT}, pre, post)$ is a tuple where:

- $E = \{e, e', \ldots\}$ is a non-empty finite set of possible events,
- $R^E_a \subseteq E \times E$ is an accessibility relation on $E$ for agent $a$,
- $pre : E \rightarrow \mathcal{L}_{EL}$ is a precondition function,
- $post : E \times AP \rightarrow \mathcal{L}_{EL}$ is a postcondition function.

A pair $(\mathcal{E}, e)$ is called an action, where $e$ represents the actual event of $(\mathcal{E}, e)$.

A pair $(\mathcal{E}, E_0)$, for $E_0 \subseteq E$, is a non-deterministic action. The set $E_0$ is the set of triggerable events.
Deterministic and non-deterministic actions

Deterministic action = single-pointed event model \((E, e)\)

\[
\begin{array}{c}
\text{pre: } p \\
\text{post: } p := q
\end{array}
\rightarrow
\begin{array}{c}
b \\
\text{pre: true} \\
\text{post: } \neg
\end{array}
\]

\[
\begin{array}{c}
a
\end{array}
\]

Non-deterministic action = multi-pointed event model

\[
\begin{array}{c}
\text{pre: true} \\
\text{post: } p := \text{true}
\end{array}
\rightarrow
\begin{array}{c}
b
\end{array}
\]

\[
\begin{array}{c}
a
\end{array}
\]

\[
\begin{array}{c}
\text{pre: true} \\
\text{post: } \neg
\end{array}
\]

\[
\begin{array}{c}
a, b
\end{array}
\]

\[
\begin{array}{c}
\text{pre: true} \\
\text{post: } p := \text{false}
\end{array}
\rightarrow
\begin{array}{c}
b
\end{array}
\]

\[
\begin{array}{c}
a
\end{array}
\]

\[
\begin{array}{c}
\text{pre: true} \\
\text{post: } \neg
\end{array}
\]

\[
\begin{array}{c}
a, b
\end{array}
\]
Public actions

Definition
An action is said to be *public* if the accessibility relations in underlying event model are self-loops.

\[
\begin{align*}
\text{pre: } & \text{true} \\
\text{post: } & p := \text{true}
\end{align*}
\]

\[
\begin{align*}
\text{pre: } & \text{true} \\
\text{post: } & p := \text{false}
\end{align*}
\]
Non-ontic actions

**Definition**
An action is said to be *non-ontic* if the postconditions are trivial: for all $e \in E$, for all propositions $p \in AP$, $\text{post}(e, p) = p$.

![Diagram](image-url)
Outline

Modeling using Dynamic Epistemic Logic (DEL)
  Epistemic states
  Epistemic languages
  Actions
  Update product
  Dynamic language
  Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Example of an update product

\[
\begin{align*}
& \text{pre: } \text{dirty}_a \\
& \text{post: } - \\
& \text{pre: } \text{true} \\
& \text{post: } -
\end{align*}
\]
**Update product: formal definition**

Let $\mathcal{M} = (W, \{R_a\}_{a \in AGT}, V)$ be an epistemic model and $\mathcal{E} = (E, (R_a^E)_{a \in AGT}, pre, post)$ be an event model.

**Definition**

The **update product** of $\mathcal{M}$ and $\mathcal{E}$ is the epistemic model $\mathcal{M} \otimes \mathcal{E} = (W^\otimes, \{R_a^\otimes\}_{a \in AGT}, V^\otimes)$ where:

$$W^\otimes = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\},$$

$$R_a^\otimes(w, e) = \{(w', e') \in W^\otimes \mid wR_aw' \text{ and } eR_a^Ee'\},$$

$$V^\otimes(w, e) = \{p \in AP \mid \mathcal{M}, w \models post(e)(p)\}.$$
Pointed update products

Definition
The successor state of an epistemic state \((M, w)\) by action \((E, e)\) is

\[(M, w) \otimes (E, e) \equiv (M \otimes E, (w, e))\]

if \(M, w \models \text{pre}(e)\), otherwise it is undefined.

Notation
- We write \(e\) instead of \((E, e)\);
- We write the word ‘\(we\)’ instead of the pair \((w, e)\);
- We write \(M \otimes E^n\) for \(M \otimes E \otimes \ldots E, n\) times.
- We write \(we_1 \ldots e_n \models \varphi\) instead of \(M \otimes E^n, we_1 \ldots e_n \models \varphi\).
Outline

Modeling using Dynamic Epistemic Logic (DEL)
- Epistemic states
- Epistemic languages
- Actions
- Update product
  - Dynamic language
  - Succinct models

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
**Dynamic language**

**Definition**

The language $\mathcal{L}_{DELCK}$ extends $\mathcal{L}_{ELCK}$ with dynamic modalities and is defined by the following BNF:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid (\varphi \lor \varphi) \mid K_a\varphi \mid C_G\varphi \mid \langle \mathcal{E}, E_0 \rangle \varphi$$

where $\mathcal{E}, E_0$ ranges over the set of non-deterministic actions.

**Definition**

We extend the definition $\mathcal{M}, w \models \varphi$ to $\mathcal{L}_{DELCK}$ with the following clause:

- $\mathcal{M}, w \models \langle \mathcal{E}, E_0 \rangle \varphi$ if there exists $e \in E_0$ s.th.
  $$\mathcal{M}, w \models pre(e) \text{ and } \mathcal{M} \otimes \mathcal{E}, (w, e) \models \varphi.$$
We define $[\mathcal{E}, E_0]$ to be $\neg \langle \mathcal{E}, E_0 \rangle \neg$.

The semantics is:

- $\mathcal{M}, w \models [\mathcal{E}, E_0] \varphi$ if for all $e \in E_0$ we have $\mathcal{M}, w \models \text{pre}(e)$ implies $\mathcal{M} \otimes \mathcal{E}, (w, e) \models \varphi$;

- $\mathcal{M}, w \models \langle \mathcal{E}, E_0 \rangle \varphi$ if there exists $e \in E_0$ s.th. $\mathcal{M}, w \models \text{pre}(e)$ and $\mathcal{M} \otimes \mathcal{E}, (w, e) \models \varphi$. 

Dual operator
Outline

Modeling using Dynamic Epistemic Logic (DEL)
  Epistemic states
  Epistemic languages
  Actions
  Update product
  Dynamic language
  **Succinct models**

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Possible world explosion

Example
Initially, number of possible worlds for Belote:

\[ \binom{32}{8} \times \binom{24}{8} \times \binom{16}{8} \approx 4 \times 10^{15} \]
Solution: succinct models

Represent succinctly epistemic and event models by:

- a Boolean formula to describe the valuations that correspond to the set of all worlds/events;
- programs (or Boolean formulas $R_a(\vec{x}, \vec{x}')$, or BDDs) for representing relations.

See [Benthem et al. 2015], [Benthem et al. 2018], [Charrier and Schwarzentruber 2017], [Charrier and Schwarzentruber 2018].
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning
  Model checking problem
  Satisfiability problem

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning
  Model checking problem
    A PSPACE procedure
    PSPACE-hardness
  Satisfiability problem

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Model checking problem

Definition

The *model checking problem* is defined as follows.

- **Input:**
  - An epistemic state $M, w$;
  - A formula $\varphi$;
- **Output:** yes if $M, w \models \varphi$; no otherwise.
Motivation: bounded epistemic planning

- Checking the existence of a bounded sequence of actions leading to a $\gamma$-state:

$$\mathcal{M}, w \models \langle \mathcal{E}, E_0 \rangle \ldots \langle \mathcal{E}, E_0 \rangle \gamma$$

iff

there are actions $e_1, \ldots e_n$ in $E_0$ such that $we_1, \ldots, e_n \models \gamma$

- Checking the existence of a bounded strategy leading to a $\gamma$-state:

$$\mathcal{M}, w \models \langle \mathcal{E}, E_0 \rangle \langle \mathcal{E}', E_0' \rangle \ldots \langle \mathcal{E}, E_0 \rangle \langle \mathcal{E}', E_0' \rangle \gamma$$
Dynamic-free language

Theorem
If $\varphi$ is dynamic-free then the model checking problem is $P$-complete.

Proof.
- P-hardness: same lower bound proof as for temporal logic CTL [Schnoebelen 2002b]
- in P: next slide
Algorithm

```
function mc(\mathcal{M}, \varphi)
    match \varphi do
        case p:
            return \{ w \mid p \text{ holds in } \mathcal{M}, w \}
        case \neg \psi:
            return \neg \psi \in mc(\mathcal{M}, \psi)
        case (\psi_1 \lor \psi_2):
            return mc(\mathcal{M}, \psi_1) \cup mc(\mathcal{M}, \psi_2)
        case K_a \psi:
            return \{ w \mid R_a(w) \subseteq mc(\mathcal{M}, \psi) \}
    check whether w \in mc(\mathcal{M}, \varphi)
```
Algorithm also for deterministic public actions

```
function mc(\(M, \varphi\))
    match \(\varphi\) do
        case \(p\) :
            return \(\{w \mid p \text{ holds in } M, w\}\)
        case \(\neg \psi\) :
            return \(mc(M, \psi)\)
        case \((\psi_1 \lor \psi_2)\) :
            return \(mc(M, \psi_1) \cup mc(M, \psi_2)\)
        case \(K_a \psi\) :
            return \(\{w \mid R_a(w) \subseteq mc(M, \psi)\}\)
        case \(\langle E, e \rangle \psi\) :
            return \(mc(M, \text{pre}(e)) \cap \{w \mid (w, e) \in mc(M \otimes E, \psi)\}\)
    end

check whether \(w \in mc(M, \varphi)\)
```
Main results

Theorem

Model checking with deterministic public actions is P-complete.

\[ \text{[van Benthem, 2011]} \]

Theorem

Model checking is PSPACE-complete.

\[ \text{[Aucher et al, 2013]} \]
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning
  Model checking problem
    A PSPACE procedure
    PSPACE-hardness
  Satisfiability problem

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
A PSPACE procedure for model checking

Specification

\[ w \tilde{e}, \varphi \rightarrow \text{mc} \rightarrow \text{yes, if } w \tilde{e} \models \varphi \]
(no otherwise)

such that \( w \tilde{e} \) is defined
A PSPACE procedure for model checking

```
function mc(w, ϕ)
    match ϕ do
        case p :
            | return inval(p, w)
        case ¬ψ :
            | return not mc(w, ϕ)
        case (ψ₁ ∨ ψ₂) :
            | return mc(M, w, ψ₁) or mc(M, w, ψ₂)
        case Kaψ :
            | for uf such that u ∈ Ra(w) and ℓ →a ℓ do
                | if in(uf) and not mc(uf, ψ) then return false
            | return true
        case ⟨E, E₀⟩ψ :
            | for e ∈ E₀ do
                | if mc(w, pre(e)) and mc(w::e, ψ) then return true
            | return false
    mc(w, ϕ)
```
Subroutines `inval` and `in`

```plaintext
function inval(p, w⃗e)
    case ⃗e = ϵ: return (p is true in w)
    case ⃗e = ⃗e′::e and: mc(w⃗e′, post(e, p))

function in(w⃗e)
    case ⃗e = ϵ: return true
    case ⃗e = ⃗e′::e: return mc(w⃗e′, pre(e)) and in(w⃗e′)
```
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Model checking problem
   A PSPACE procedure
   PSPACE-hardness
Satisfiability problem

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
PSPACE-hardness

Theorem

Model checking is PSPACE-hard.

Proof.

\[ \exists p \forall q \ldots \psi \rightarrow \text{reduction} \rightarrow M, w, \varphi \rightarrow \text{model checking} \rightarrow \text{yes/no} \]

\[ \varphi := \langle p := \text{false} \cup p := \text{true} \rangle[q := \text{false} \cup q := \text{true}] \ldots \psi \]
PSPACE-hardness

Theorem

Model checking is PSPACE-hard already for:

- Non-deterministic public actions (previous slide);

Further reading: parameterized complexity for DEL model checking: Pol, Rooij, and Szymanik 2015

<table>
<thead>
<tr>
<th>Deterministic public actions</th>
<th>Explicit models</th>
<th>Succinct models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic public actions</td>
<td>P-c</td>
<td>PSPACE-c</td>
</tr>
<tr>
<td>All</td>
<td>PSPACE-c</td>
<td>PSPACE-c</td>
</tr>
</tbody>
</table>
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning
   Model checking problem
   Satisfiability problem

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Satisfiability problem definition

Definition
The satisfiability problem in DEL is the following decision problem.

- Input: a formula $\varphi$;
- Output: yes if there is an epistemic state $\mathcal{M}, w$ such that $\mathcal{M}, w \models \varphi$; no otherwise.
Motivation: parameterized bounded epistemic planning

- there exists a bounded sequence of actions leading to a $\gamma$-state from any $\psi$-epistemic state iff $\psi \rightarrow \langle E, E_0 \rangle \ldots \langle E, E_0 \rangle \gamma$ is satisfiable

- There is a bounded strategy leading to a $\gamma$-state from any $\psi$-epistemic state: iff $\psi \rightarrow \langle E, E_0 \rangle \langle E', E'_0 \rangle \ldots \langle E, E_0 \rangle \langle E', E'_0 \rangle \gamma$ is satisfiable
Complexity results

EL  
\( \text{mc: P-c} \)  
\( \text{sat: PSPACE-c} \)  
[Schnoebelen 2002a]

ELCK  
\( \text{mc: P-c} \)  
\( \text{sat: EXPTIME-c} \)  
[Schnoebelen 2002a],  
[Halpern and Moses 1992]

DEL  
\( \text{mc: PSPACE-c} \)  
\( \text{sat: NEXPTIME-c} \)  
[Aucher and __, 2013],  
[Bolander, Jensen and __, 2015b],  
[Pol, Rooij, and Szymanik 2015]

DELCK  
\( \text{mc: PSPACE-c} \)  
\( \text{sat: 2EXPTIME-c} \)  
[Charrier and __, 2018]

All complexities remain the same for succinct event models in the language, except P-c becomes PSPACE-c (see [Charrier and __, 2018]).
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning
  Epistemic planning problem
  Planning as a first-order query in DEL structures
  Undecidability
  Event model restrictions

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Epistemic planning problem
  Planning as a first-order query in DEL structures
  Undecidability
  Event model restrictions

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Epistemic planning instance

Definition
An epistemic planning instance is a tuple $\mathcal{M}, w, \mathcal{E}, \mathcal{E}_0, \gamma$ where:

- $\mathcal{M}, w$ is a pointed epistemic model; (initial situation)
- $\mathcal{E}$ is an event model;
- $\mathcal{E}_0$ is a subset of events in $\mathcal{E}$; (repertoire of events)
- $\gamma$ an epistemic formula. (goal)
Example of planning instance \((\mathcal{M}, w, \mathcal{E}, E_0, \gamma)\):
Epistemic planning problem

Definition

The epistemic planning problem is defined as follows:

- **Input**: an epistemic planning instance \((\mathcal{M}, w, \mathcal{E}, E_0, \gamma)\);

- **Output**: yes if there exists a sequence \(e_1, \ldots, e_\ell \in E_0\) such that \(we_1 \ldots e_\ell \models \gamma\); no otherwise.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning
  Epistemic planning problem
  Planning as a first-order query in DEL structures
  Undecidability
  Event model restrictions

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Planning as a first-order query in DEL structures
Planning as a first-order query in DEL structures
Deliberation Dynamics and Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Epistemic planning problem

Planning as a first-order query in DEL structures

Undecidability

Event model restrictions

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion

References

DEL presentation: formal definition

Definition

A DEL presentation is a pair \((M, E)\) where \(M\) is an epistemic model and \(E\) is an event model.

Let \(M = (W, (R_a)_{a \in AGT}, V)\) be an epistemic model and \(E = (E, (R^E_a)_{a \in AGT}, pre, post)\) be an event model.

Notation

- \(\mathcal{H}_n\) is the set of worlds of \(M \otimes E^n\).

- Worlds of \(M \otimes E^n\) are written \(h = we_1 \ldots e_n\).
DEL structure: formal definition

Let $(\mathcal{M}, \mathcal{E})$ be a DEL presentation. A DEL structure is the unraveling of some DEL presentation $(\mathcal{M}, \mathcal{E})$.

Definition
The DEL structure denoted by $(\mathcal{M}, \mathcal{E})$ is the structure

$$\mathcal{M}\mathcal{E}^* = (\mathcal{H}, \rightarrow, (R_a)_{a \in AGT}, (p)_{p \in AP}),$$

where

- $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$; (histories)
- $h \rightarrow h'$ whenever $h' = he$ for some event $e$;
- $hR_a h'$ whenever $hR_a h'$ in $\mathcal{M} \otimes \mathcal{E}^n$, for some $n$;
- $p(h)$ holds if $p$ holds in $h$ in $\mathcal{M} \otimes \mathcal{E}^n$. 


Epistemic logic embedded in First-order logic

**Theorem**

*Given an epistemic formula* $\gamma$, *one can effectively compute a first-order formula* $\text{tr}(\gamma)(x)$ *such that*

$$\mathcal{ME}^*, h \models \gamma \text{ iff } \mathcal{ME}^*, [x := h] \models \text{tr}(\gamma)(x).$$

**Example**

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$\text{tr}(\gamma)(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_a p$</td>
<td>$\forall y R_a(x, y) \rightarrow p(y)$</td>
</tr>
<tr>
<td>$q \land \hat{K}_a q$</td>
<td>$q(x) \land \exists y R_a(x, y) \land q(y)$</td>
</tr>
</tbody>
</table>
Planning as a first-order query

Proposition

A planning instance $\mathcal{M}, w, \mathcal{E}, E_0, \gamma$ is positive

iff there exists a history $we_1 \ldots e_\ell$ of $\mathcal{M}\mathcal{E}^*$ such that:

- $e_1, \ldots, e_\ell \in E_0$;
- $we_1 \ldots e_\ell \models \gamma$;

iff $\mathcal{M}\mathcal{E}^* \models \exists \! x (\text{history}_{E_0}(x) \land \text{tr}(\gamma)(x))$

PS: handling $\text{history}_{E_0}(x)$ is small technical detail...
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning
  Epistemic planning problem
  Planning as a first-order query in DEL structures
  Undecidability
  Event model restrictions

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Undecidability of epistemic planning

Theorem

*Epistemic planning problem is undecidable.*

Proof.

DEL structures are Turing-complete! ([Bolander and Andersen 2011], [Cong, Pinchinat, and Schwarzentruber 2018])
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

**Unbounded epistemic planning**
- Epistemic planning problem
- Planning as a first-order query in DEL structures
- Undecidability

**Event model restrictions**

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Event model restrictions

Modal depth

\[ K_a K_b p: \quad md = 2 \]
\[ K_a \hat{K}_b \hat{K}_c p: \quad md = 3 \]

<table>
<thead>
<tr>
<th></th>
<th>pre</th>
<th>(md = 0)</th>
<th>(md = 1)</th>
<th>(md \geq 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>post</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non-ontic</td>
<td>dec</td>
<td>?</td>
<td>undec</td>
<td></td>
</tr>
<tr>
<td>ontic</td>
<td>dec</td>
<td></td>
<td>undec</td>
<td></td>
</tr>
</tbody>
</table>

- What we just seen
- Similar proof (see [Aucher and Bolander 2013], [Charrier, Maubert, and Schwarzentruber 2016])
- Open problem
- Next section!
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning
Automatic structures
PDEL structures are automatic
Wrap up

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post
  PDEL Planning
    Automatic structures
    PDEL structures are automatic
    Wrap up

Knowledge-based programs

Conclusion
PDEL Planning

Call **PDEL presentation** a DEL presentation where every precondition is propositional, and call **PDEL structure** a DEL structure arising from a PDEL presentation.

**Definition (PDEL planning)**

- **Input:** an epistemic planning instance \((M, w, \mathcal{E}, E_0, \varphi)\) where \((M, \mathcal{E})\) is a PDEL presentation;
- **Output:** yes if there exists a history \(we_1 \ldots e_\ell\) in \(M\mathcal{E}^*\) such that \(we_1 \ldots e_\ell \models \varphi\) and \(e_1, \ldots, e_\ell \in E_0\).
Is PDEL planning decidable?
Issue: the DEL structure is infinite...

Two possible attitudes towards infinite objects

- Try to prove Turing-completeness hence undecidability;
- Try to prove regularity of the structure hence decidability.
Theorem

PDEL planning is decidable ([Yu, Wen, and Liu 2013], [Aucher, Maubert, and Pinchinat 2014]).

Proof.

DEL planning is a FO-query

FO-query on automatic structures is decidable.

PDEL structures are automatic

It is even decidable for epistemic linear μ-calculus!

[Douéneau-Tabot, Pinchinat, and Schwarzentruber 2018]
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning

Automatic structures
  Finite automata
  Automatic presentations
  First-order logic on automatic structures

PDEL structures are automatic

Wrap up

Knowledge-based programs

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning

Automatic structures

Finite automata

Automatic presentations

First-order logic on automatic structures

PDEL structures are automatic

Wrap up

Knowledge-based programs

Conclusion
Finite automata

Let $\Sigma$ be an alphabet. $\Sigma^*$ is the set of all finite words over $\Sigma$.

Definition

A word automaton $A$ is a tuple $A = (S, \iota, \Delta, F)$ where

- $S$ is a finite set of states, $\iota \in S$ is the initial state;
- $\Delta \subseteq S \times \Sigma \times S$ is the transition relation;
- $F \subseteq S$ is the set of accepting states.

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\ Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$

$\Sigma$
Regular languages

- An execution of $A$ on $\alpha = \ell_1 \ldots \ell_n \in \Sigma^*$...
- A word is accepted by $A$ if there exists an accepting execution of $A$ on it.
- The language accepted by $A$ is the set $L(A) \subseteq \Sigma^*$ of all words accepted by $A$.

Definition

A language $L \subseteq \Sigma^*$ is **regular** if there exists a finite automaton $A$ such that $L = L(A)$.

The language accepted by the automaton drawn above is the set of words of the form $01 \ldots 10$, and is often written $01^*0$.

Theorem

*The emptiness problem for word automata is decidable in $\mathbb{N} \logspace$.*
Regular relations

Let $\Sigma_\bot = \Sigma \cup \{\bot\}$, where $\bot$ is a fresh symbol.

\begin{center}
\begin{tabular}{c|c|c|c|c}
& $\ell_1^1$ & $\ell_2^1$ & $\ell_3^1$ & $\bot$ \\
$\eta_1$ & $\ell_1$ & $\ell_2$ & $\ell_3$ & $\bot$ \\
$\eta_2$ & $\ell_1^2$ & $\ell_2^2$ & $\ell_3^2$ & $\ell_4^2$ \\
$\eta_3$ & $\ell_1^3$ & $\ell_2^3$ & $\bot$ & $\bot$ \\
\end{tabular}
\end{center}

$A$ \rightarrow \text{yes/no}

Definition

The convolution of $\eta_1, \ldots, \eta_n \in \Sigma^*$, written $\odot(\eta_1, \ldots, \eta_n)$, is the word over alphabet $(\Sigma_\bot)^n$ obtained by left-aligning $\eta_1, \ldots, \eta_n$ while completing with $\bot$.

Definition

The convolution of a relation $R \subseteq (\Sigma^*)^n$ is the language

$$\odot R = \{\odot(\eta_1, \ldots, \eta_n) \mid (\eta_1, \ldots, \eta_n) \in R\} \subseteq ((\Sigma_\bot)^n)^*$$

Definition

$R \subseteq (\Sigma^*)^n$ is regular whenever there is a finite automaton over alphabet $(\Sigma_\bot)^n$ that accepts $\odot R$. 
Examples of regular relations

- The binary equal-length relation \( el \), i.e., pairs \((\eta, \eta')\) with \(|\eta| = |\eta'|\).

- The binary prefix relation \( \preceq \).

\[
\begin{align*}
\text{start} & \rightarrow l \\
(\ell) & \rightarrow (\ell') \\
l & \rightarrow l
\end{align*}
\]

\[
\begin{align*}
\text{start} & \rightarrow l \\
(\ell) & \rightarrow (\ell) \\
l & \rightarrow (\bot) \\
(\bot) & \rightarrow (\bot)
\end{align*}
\]
Closure properties of regular relations

Theorem

Let $R, R'$ be regular relations over $\Sigma^*$. Then the following relations are also regular:

- **Union** $R \cup R'$;
- **Intersection** $R \cap R'$;
- **Relative complementation** $R \setminus R'$;

Moreover there is an effective procedure that, given automata for $\circ R$ and $\circ R'$, computes an automaton for the convolution of each of the resulting relations.

Proof.

Use standard automata constructions, e.g., synchronous product for intersection.

Remark

Computing the automaton for $\circ R \setminus R'$ requires to complement $A$ for $\circ R'$, that relies on the determinization of $A$. (an exponential cost in general; it is a powerset construction).
The projection of a regular relation is regular

**Theorem**

Let $R \subseteq (\Sigma^*)^r$ be regular relation.
Then one can effectively compute an automaton $B$ s.t.

$$L(B) = \bigcirc(\{(\eta_2, \ldots, \eta_r)\mid \text{there exists } \eta_1, (\eta_1, \eta_2, \ldots, \eta_r) \in R\}).$$

**Proof.**
Forget the first coordinate.

**Example**

$$\begin{align*}
(\varepsilon), (f), (g), (g), (f)
\end{align*}$$

**Remark**

The projected automaton is non-deterministic in general.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning
  Automatic structures
    Finite automata
    Automatic presentations
    First-order logic on automatic structures

PDEL structures are automatic

Wrap up

Knowledge-based programs

Conclusion

References
### Automatic presentations

Let $\mathcal{S} = \langle S, (R_i)_{i \in I} \rangle$ be a structure.

**Definition**

An **automatic presentation** of $\mathcal{S}$ consists of a pair $(\bar{A}, \nu)$ s.t.

- $\bar{A}$ is a tuple of automata $\langle A_S, (A_{R_i})_{i \in I} \rangle$;
- $\nu : L(A_S) \rightarrow S$ is a bijective mapping, and we let

$$\nu^{-1}(R_i) := \{(\eta_1, \ldots, \eta_{r_i}) \in (\Sigma^*)^{r_i} | R_i(\nu(\eta_1), \ldots, \nu(\eta_{r_i}))\}.$$  

s.t. $L(A_{R_i}) = \circ \nu^{-1}(R_i)$.

Intuitively, words from $L(A_S)$ encode elements of $S$ (via mapping $\nu$) in such a way that the induced relations $\nu^{-1}(R_i)$ are regular.

An **automatic structure** is a structure that has an automatic presentation.

### Example

$(\mathbb{N}, succ)$ with $succ = \{(n, n+1) | n \in \mathbb{N}\}$ is an automatic structure: take alphabet $\Sigma = \{\ell\}$ and $\nu : \ell^* \rightarrow \mathbb{N}$, and automaton for relation $\circ succ$ is the one for words of the form $\ell \ell \ldots \ell \perp \ell$. 

---

**Note:** The text snippet includes a table with rows labeled as follows:

- Modelung using Dynamic Epistemic Logic (DEL)
- Bounded epistemic planning
- Unbounded epistemic planning
- Automatic structures for decidability of unbounded epistemic planning when propositional pre/post PDEL Planning
- Automatic structures Finite automata
- **Automatic presentations**
- First-order logic on automatic structures
- PDEL structures are automatic
- Wrap up
- Knowledge-based programs
- Conclusion
- References

---

**Page Number:** 80
Other examples of automatic structures

- Every finite structure is automatic.
- Given a DEL presentation where pre/post are propositional, the associated DEL structure is automatic. (next section)
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning

Automatic structures
- Finite automata
- Automatic presentations
- First-order logic on automatic structures

PDEL structures are automatic

Wrap up

Knowledge-based programs

Conclusion
First-order logic on automatic structures

**Theorem**

*For every automatic presentation* \((\bar{A}, \nu)\) *of structure, every first-order formula* \(\Phi(x_1, \ldots, x_n)\) *induces a relation* \(R\) *of arity* \(n\) *with* \(\nu^{-1}(R)\) *regular. Moreover, the automaton for* \(\circ \nu^{-1}(R)\) *can be effectively computed.*

**Bottom-up construction:**

1. Project \(A_{R_2(z,x)}\) on first component and get \(A_{\exists z R_2(x,z)}\);
2. Complement \(A_{p(x)}\), get \(A_{c p(x)}\), compute \(A_S \cap A_{c p(x)}\) and get \(A_{\neg p(x)}\);
3. Compute \(A_{\exists z R_2(x,z)} \cap A_{\neg p(x)}\) to get \(A_{\exists z R_2(z,x) \land \neg p(x)}\).*
First-order logic on automatic structures

**Theorem**

*For every automatic presentation* $(\tilde{A}, \nu)$ *of structure, every first-order formula* $\Phi(x_1, \ldots, x_n)$ *induces a relation* $R$ *of arity* $n$ *with* $\nu^{-1}(R)$ *regular. Moreover, the automaton for* $\circ \nu^{-1}(R)$ *can be effectively computed.*

**Bottom-up construction:**

Take $\exists z R_2(z, x) \land \neg p(x)$.

1. Project $A_{R_2(z, x)}$ on first component and get $A_{\exists z R_2(x, z)}$;
2. Complement $A_{p(x)}$, get $A_{c p(x)}$, compute $A_{S} \cap A_{c p(x)}$ and get $A_{\neg p(x)}$;
3. Compute $A_{\exists z R_2(x, z) \cap A_{\neg p(x)}}$ to get $A_{\exists z R_2(z, x) \land \neg p(x)}$.

**Corollary**

*The first-order theory of each automatically presentable structure is decidable.*
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning
Automatic structures
PDEL structures are automatic
Wrap up

Knowledge-based programs

Conclusion
Theorem

Given a PDEL presentation \((\mathcal{M}, \mathcal{E})\), the structure \(\mathcal{M}\mathcal{E}^* = (H, \rightarrow, (R_a)_{a \in AGT}, (p)_{p \in AP})\) is automatic.

Proof: We exhibit an automatic presentation \((\bar{A}, \nu)\).

First, \(\nu := id\), that is, every history \(we_1 \ldots e_n \in H\) is encoded as the word \(we_1 \ldots e_n \in (W \cup E)^*\).

Now we define \(\bar{A} = \langle A_H, A_\rightarrow, (A_{R_a})_{a \in AGT}, (A_p)_{p \in AP} \rangle\).
Some ideas for $A_{\mathcal{H}}$

Notation

- *Given an event e, view pre($e$) as a subset of valuations.*
  
  e.g., view $p \lor q$ as $\{\{p\}, \{q\}, \{p, q\}\}$.

- *For all valuations $P$, let $P \otimes post(e)$ be the valuation $P$ updated by post($e$)*
  
  e.g., $\{p, q\} \otimes [p := \bot, r := \top] = \{q, r\}$.

Idea for $A_{\mathcal{H}}$:

$$L(A_{\mathcal{H}}) = \{w_1, w_2, w_3, \ldots, w_1e, \ldots, w_1ee', \ldots\}$$
Definition of $\mathcal{A}_H$, and of $\mathcal{A}_p \ (p \in AP)$

Let $\mathcal{A}_H = (S, \iota, \Delta, S \setminus \\{\iota\})$ where

- $S = \{\iota\} \cup 2^{AP}$;
- $(\iota, w, V(w)) \in \Delta$, for every $w \in W$;
- $(P, e, P \otimes post(e)) \in \delta$ whenever $P \in \text{pre}(e)$.

Incidentally, we take $\mathcal{A}_p = (S, \iota, \Delta, \{P \mid p \in P\})$. 
Definition of $A_\rightarrow$

We want an automaton for

$$\bigcirc(\rightarrow) = \left\{ (\begin{array}{c} u \\ u \end{array}) \ldots (\begin{array}{c} e_n \\ e_n \end{array}) (\begin{array}{c} \bot \\ e_{n+1} \end{array}) \mid ue_1 \ldots e_ne_{n+1} \in \mathcal{H} \right\}$$

- First, consider $A$:

- Second, we make sure that accepted pairs are histories. Build automaton $B$ for the binary relation $\mathcal{H} \times \mathcal{H}$ and define:

$$A_\rightarrow = A \cap B$$
Definition of $A_{Ra}$

$$A_{Ra} = A \cap B$$

where $A$ is:

$$(w, u), wR_{Ra}u$$

and automaton $B$ is as previous slide before for $H \times H$.

This ends the proof of Theorem on Slide 85.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

PDEL Planning
Automatic structures
PDEL structures are automatic
Wrap up

Knowledge-based programs

Conclusion
Epistemic planning: a view on the DEL structure

- **Input:** an epistemic planning instance \((\mathcal{M}, w, \mathcal{E}, E_0, \varphi)\) where \((\mathcal{M}, \mathcal{E})\) is a PDEL presentation;
- **Output:** yes if there exists a history \(we_1 \ldots e_\ell\) in \(\mathcal{M}\mathcal{E}^*\) such that \(we_1 \ldots e_\ell \models \varphi\) and \(e_1, \ldots, e_\ell \in E_0\).

Amounts to query \(\mathcal{M}\mathcal{E}^* \models \exists x\,(\text{historyE0}(x) \land tr(\gamma)(x))\)
Decidability of propositional epistemic planning

- **Input:** an epistemic planning instance \((\mathcal{M}, w, \mathcal{E}, E_0, \varphi)\) where \((\mathcal{M}, \mathcal{E})\) is a PDEL presentation;
- **Output:** yes if there exists a history \(w e_1 \ldots e_\ell\) in \(\mathcal{M}\mathcal{E}^*\) such that \(w e_1 \ldots e_\ell \models \varphi\) and \(e_1, \ldots, e_\ell \in E_0\).

Ex: \(\gamma = K_a\hat{K}_b p\).

Amounts to query \(\mathcal{M}\mathcal{E}^* \models \exists x (\text{historyE0}(x) \land tr(\gamma)(x))\).

Sketch of an algorithm:

1. (For predicate \text{historyE0}) Take \(A_{\text{historyE0}}\) that accepts all words \(w e_1 \ldots e_n\) with \(e_1, \ldots, e_n \in E_0\);
2. Compute \(A_{tr(\gamma)}\);
   Ex: \(tr(\gamma)(x) = \forall y[R_a(x, y) \rightarrow \exists z(R_b(y, z) \land p(z))]\).
   \(L(A_{tr(\gamma)}) = \{h \mid \mathcal{M}\mathcal{E}^*, [x := h] \models tr(\gamma)(x)\}\).
3. Compute \(A\) s.t. \(L(A) = L(A_{\text{historyE0}}) \cap L(A_{tr(\gamma)})\)
4. Return “yes” if \(L(A) \neq \emptyset\), “no” otherwise.
Propositional epistemic plan synthesis

Since \(\nu : L(A_H) \rightarrow \mathcal{H}\) is the identity mapping, i.e., \(\nu^{-1}(h) = h\), we can synthesize the set of successful plans for \(\gamma\).

**Theorem**

Let \(A\) be the automaton for \(\text{history}E_0(x) \land \text{tr}(\gamma)(x)\). Then \(L(A)\) contains exactly all words/histories we\(_1\) \(\ldots\) e\(_\ell\) s.t.

- \(e_1, \ldots, e_\ell \in E_0\);
- \(M\mathcal{E}^*\), we\(_1\) \(\ldots\) e\(_\ell\) \(\models\) \(\gamma\).

**Corollary**

Let \((M, w, \mathcal{E}, E_0, \varphi)\) be an instance of PDEL planning problem. We can effectively construct an automaton accepting the set of successful plans, i.e., sequences e\(_1\) \(\ldots\) e\(_\ell\) \(\in E_0^*\) such that

\[M\mathcal{E}^*, \text{we}_1 \ldots \text{e}_\ell \models \gamma\]
Complexity of PDEL planning

That is of the query $\mathcal{M} \mathcal{E}^* \models \exists x (\text{historyE0}(x) \land \text{tr}(\gamma)(x))$.

- The complexity is at most $d\text{-EXPTIME}$ where $d$ is the alternation depth of $\exists x (\text{historyE0}(x) \land \text{tr}(\gamma)(x))$.

E.g. take $\exists x \forall y \exists z R(x, y, z)$, which is $\exists x \neg \exists y \neg \exists z R(x, y, z)$.

To build the automaton for $\neg \psi$, one needs to complement $A_\psi$. Since $A_\psi$ may result from projection operations, it may involve a determinization, hence an exponential blow up.

- The lower bound complexity of the PDEL planning is unknown.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
  Mathematical Properties
  Succinctness
  Conclusion

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Motivation
Syntax of Knowledge-based programs
Semantics
Mathematical Properties
Succinctness
Conclusion

Conclusion
Automation of complex tasks

- Building surveillance
- Nuclear decommissioning
- Intelligent farming
Multiple robots

more robust/efficient than
Multiple robots

more robust/efficient than
Multiple robots

more robust/efficient than
Multiple robots

more robust/efficient than

Settings

- Cooperative agents;
- Common goal;
- Imperfect information;
Methodology

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post Knowledge-based programs

Motivation

Syntax of Knowledge-based programs

Semantics

Mathematical Properties

Succinctness

Conclusion

References
Need: understandable system

Motivation

- Legal issues in case of failure
- Interaction with humans

```c
#include "fixed.h"
#include "fixed_private.h"

int64_t error;
int64_t torque_request;
DWork DWork;

void fixed_step(void)
{
    int64_t FilterCoefficient_m = ((int64_t)(int32_t)((int32_t)(5403L * (int32_t)error >> 1U) - DWork.Filter_DSTATE) << 4U) + 17599L >> 14);
    torque_request = (((int64_t)(12475L * (int32_t)error >> 14U) >> 1) + (DWork.Integrator_DSTATE >> 2) + (FilterCoefficient_m >> 1));
    DWork.Integrator_DSTATE = ((int64_t)((4683L * (int32_t)error >> 15U) * 5248L >> 16U) + DWork.Integrator_DSTATE);
    DWork.Filter_DSTATE = (int64_t)(5248L * (int32_t)FilterCoefficient_m >> 16U) + DWork.Filter_DSTATE;
}

void fixed_initialize(void)
{
    torque_request = 0;
    (void) memset((void *)&DWork, 0,
    sizeof(DWork));
    error = 0;
}
```
Advertising: use of knowledge-based programs


KBP for agent $a$

- listenRadio
- if $K_a$strike
  - toStation
- else
  - toAirport

KBP for agent $b$

- readNewsPaper
- if $K_b$strike
  - toStation
- else
  - toAirport

- Understand coordination of agents in QdecPOMDP;
- Succinctness;
- (-) (Un)decidability/complexity issues.

Recent work [Saffidine, Schwarzentruber, and Zanuttini 2018] that extends the mono-agent case in [Lang and Zanuttini 2012], [Lang and Zanuttini 2013].
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
  Mathematical Properties
  Succinctness
  Conclusion

Conclusion
Properties expressed in epistemic logic

Language constructions

room 43 is safe  door 12 is locked  justobserved(🔥)  . . .

¬...
(... ∨ ...)
(... ∧ ...)
(... → ...)
(K... ...)

Example

(K_a  door 12 is locked) ∧ ¬(K_c  door 12 is locked)
K_a(K_c  door 12 is locked) ∨ K_a¬(K_c  door 12 is locked)
Program constructions

Language constructions

turn left    stay    broadcast temperature

..., ...

if $\varphi$ then ...else ...

while $\varphi$ do ...

Example (knowledge-based program for agent $a$)

```
if $K_a(\text{door 12 is locked} \land justobserved(\text{fire}))$ then
  turn left
  broadcast temperature
else
  stay
```
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Motivation

Syntax of Knowledge-based programs

Semantics
  Models: QdecPOMDP
  Operational semantics of KBPs

Mathematical Properties

Succinctness

Conclusion
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Motivation

Syntax of Knowledge-based programs

Semantics
  Models: QdecPOMDP
  Operational semantics of KBPs

Mathematical Properties

Succinctness

Conclusion
QdecPOMDP

Qualitative decentralized Partially Observable Markov Decision Processes
= Concurrent game structures with observations.

Transitions of the form:

\[ \begin{align*}
  a & : \text{stay} & a & : \text{burn} \\
  b & : \text{turn left} & b & : \text{obscure} \\
\end{align*} \]

state1 \[\rightarrow\] state2

A non-empty set of possible initial states;

A set of goal states.
States

Typically, a state describes:

- positions of agents;
- battery levels;
- etc.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
    Models: QdecPOMDP
    Operational semantics of KBPs
  Mathematical Properties
  Succinctness
  Conclusion
Operational semantics

Epistemic structure

Higher-order knowledge about:

- the current state of the QdecPOMDP;
- the current program counters in KBPs.
Assumptions

Common knowledge of:
- the QdecPOMDP;
- the KBPs;
- synchronicity of the system;
  - tests last 0 unit of time;
  - actions last 1 unit of time.

KBP for agent $a$

\[
\begin{array}{ll}
\text{listenRadio} & \\
\text{if } K_a \text{strike} & \\
\quad \text{toStation} & \\
\text{else} & \\
\quad \text{toAirport} & 
\end{array}
\]

KBP for agent $b$

\[
\begin{array}{ll}
\text{readNewsPaper} & \\
\text{if } K_b \text{strike} & \\
\quad \text{toStation} & \\
\text{else} & \\
\quad \text{toAirport} & 
\end{array}
\]
Epistemic structures at time $T$: worlds

Worlds = consistent histories of the form

\[ s^0 \xrightarrow{pC^0} obs^1 s^1 \xrightarrow{pC^1} \ldots \xrightarrow{obs^T s^T pC^T} \]

where

- $\xrightarrow{obs^t}$ vector of observations at time $t$
- $s^t$ state at time $t$
- $\xrightarrow{pC^t}$ vector of program counters at time $t$
Epistemic structures at time $t$: indistinguishability relations

Agent $a$ confuses two histories iff she has received the same observations.

\[
\begin{align*}
s^0 p c^0 & \rightarrow obs^1 s^1 p c^1 \ldots obs^T s^T p c^T \\
\rightarrow obs' s' p c' & \rightarrow \ldots \rightarrow obs'_T s'_T p c'_T
\end{align*}
\]

iff for all $t \in \{1, \ldots, T\}$, $obs^t_a = obs'_t_a$
Program counters

Definition (Program counter)
(guard, action just executed, continuation)

- listenRadio
- if $K_a\text{strike}$ then
  - toStation
else
  - toAirport

$(\top, \text{start}, \bullet)$
$(\top, \text{listenRadio}, \blacksquare)$
$(K_a\text{strike}, \text{toStation}, \bigtriangledown)$
$(\neg K_a\text{strike}, \text{toAirport}, \bigtriangledown)$
Control-flow graph

- listenRadio
- if $K_a\text{strike}$ then
  - toStation
else
  - toAirport

$(\top, \text{start}, \bullet)$

$(\top, \text{listenRadio}, \Box)$

$(K_a\text{strike}, \text{toStation}, \bigtriangleup)$

$(\neg K_a\text{strike}, \text{toAirport}, \bigtriangleup)$
Consistent histories (explained with one agent)

In the QdecPOMDP:

\[ s^0 \xrightarrow{\text{listenRadio}} s^1 \]
\[ s^1 \xrightarrow{\text{toStation}} s^2 \]

KBP control-flow graph

\[ (\top, \text{start}, \bullet) \]
\[ (\top, \text{listenRadio}, \blacksquare) \]
\[ (K_a\text{strike}, \text{toStation}, \blacktriangledown) \]
\[ (\neg K_a\text{strike}, \text{toAirport}, \blacktriangledown) \]

\[ s^0 (\top, \text{start}, \bullet) s^1 (\top, \text{listenRadio}, \blacksquare) s^2 (K_a\text{strike}, \text{toStation}, \blacktriangledown) \]

\[ \models K_a\text{strike} \]
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
  Mathematical Properties
    Verification
    Execution Problem

Succinctness

Conclusion

References
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Motivation

Syntax of Knowledge-based programs

Semantics

Mathematical Properties

Verification

Execution Problem

Succinctness

Conclusion

References
Verification problem

Definition

Input:
- A QdecPOMDP model (given in STRIPS-like symbolic form);
- Knowledge-based programs for each agent;

Output: yes if all executions of the KBPs lead to a goal state.

Theorem

The verification problem for while-free KBPs is PSPACE-complete, and is undecidable for general KBPs.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
  Mathematical Properties
    Verification
    Execution Problem
  Succinctness
  Conclusion

References
Execution Problem

Input:
- an agent $a$;
- a QdecPOMDP model;
- policies (e.g. KBPs), one for each agent;
- a local view of the history for agent $a$.

Output: the action $act$ agent $a$ should take.
Execution Problem  (decision problem)

Input:

- an agent $a$;
- a QdecPOMDP model;
- policies (e.g. KBPs), one for each agent;
- a local view of the history for agent $a$;
- an action $act$.

Output: yes, if the next action of agent $a$ is $act$; no otherwise.
Reactive policy representation

Definition (reactive policy representation)
A class of policy representations is reactive
iff its corresponding execution problem is in P.

Example (Tree policies are reactive policy representation)
\[
\text{if } \text{justobserved}(\text{fire}) \text{ then } \text{turn left} \text{ else } \text{stay}
\]

Unless P = PSPACE, KBPs are not reactive. Indeed:

Proposition
The execution problem for KBPs is PSPACE-complete.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
    Motivation
    Syntax of Knowledge-based programs
    Semantics
    Mathematical Properties
    Succinctness
    Conclusion

Conclusion
Modal depth

Modal depth = number of nested ‘$K...$’ operators.

<table>
<thead>
<tr>
<th>Formulas</th>
<th>Modal depths</th>
</tr>
</thead>
<tbody>
<tr>
<td>$justobserved($)</td>
<td>0</td>
</tr>
<tr>
<td>$K_a p$</td>
<td>1</td>
</tr>
<tr>
<td>$K_a(K_b p)$</td>
<td>2</td>
</tr>
</tbody>
</table>
Theorem ([Lang, Zanuttini, 2012] for $d = 1$; [AAAI2018], for $d > 1$)

Let $d \geq 1$.

There is a poly($n$)-size $Q\text{decPOMDP}$ family $(M_{n,d})_{n \in \mathbb{N}}$ for which:

1. there is a $d$-modal depth poly($n$)-size valid KBP family;
2. no $(d - 1)$-modal depth valid KBP family;
3. assuming $NP \not\subseteq P/poly$, for any reactive policy representations, no poly($n$)-size valid policy family.
Succinctness

Theorem ([Lang, Zanuttini, 2012] for $d = 1$; [AAAI2018], for $d > 1$)
Let $d \geq 1$.
There is a $\text{poly}(n)$-size $\text{QdecPOMDP}$ family $(M_{n,d})_{n \in \mathbb{N}}$ for which:

1. there is a $d$-modal depth $\text{poly}(n)$-size valid $\text{KBP}$ family;
2. no $(d - 1)$-modal depth valid $\text{KBP}$ family;
3. assuming $\text{NP} \not\subseteq \text{P/poly}$, for any reactive policy representations, no $\text{poly}(n)$-size valid policy family.

Proof idea. $M_{n,d}$:
- run a $\text{poly}(n)$-time protocol revealing a $\text{poly}(n)$-size $3$-CNF $\beta$;
- $\beta$ satisfiable iff a $d$-md non $d - 1$-md expressible epistemic property holds.
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs
  Motivation
  Syntax of Knowledge-based programs
  Semantics
  Mathematical Properties
  Succinctness
  Conclusion

Conclusion
Conclusion

Higher-order knowledge...

- for get explainable policies (e.g. making cooperation visible)
- for concise programs
Outline

Modeling using Dynamic Epistemic Logic (DEL)

Bounded epistemic planning

Unbounded epistemic planning

Automatic structures for decidability of unbounded epistemic planning when propositional pre/post

Knowledge-based programs

Conclusion
Perspectives

- Design efficient implementation for PSPACE problems;
- Extend algorithms with probabilities;
- Learn policies that are knowledge-based policies;
- Limited beliefs: more efficient and natural behaviors.
Acknowledgment

- Sophie Pinchinat: the slides about automatic structures
- Tristan Charrier, PhD Student, many complexity results in succinct models, model checking, SAT
- Gaëtan Douenot-Tabot: his work in automata theory
- Hans van Ditmarsch, Valentin Goranko, Andreas Herzig, Emiliano Lorini, Thomas Bolander, Abdallah Saffidine, Bruno Zanuttini, etc.
- The School of Art LISAA, Rennes, for the design of Hintikka’s world
- Anass Lakhar, Eva Soulier: students contributing to Hintikka’s world


Iris van de Pol, Iris van Rooij, and Jakub Szymanik. “Parameterized Complexity Results for a Model of Theory of Mind Based on Dynamic Epistemic Logic”. In: *Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015, Carnegie Mellon University, Pittsburgh, USA, June 4-6, 2015*. 2015, pp. 246–263. DOI: 10.4204/EPTCS.215.18. URL: https://doi.org/10.4204/EPTCS.215.18.


