Towards an efficient representation for epistemic planning
Supervised by Alexandre Niveau

Sebastien Gamblin
University of Caen

December 6, 2018
Plan

1. Context of the work

2. Theory
   - Epistemic logic
   - Kripke structure
   - Product update
   - KBP

3. Work
   - V-injectivity
   - Propositional representation
   - Simple querying of the structure
   - Planification
   - Results

4. Conclusion, perspectives
Theme: Multi-agent planification, using epistemic logic and event model of DEL to represent the problem.

Goal: We want to find one policy for each agent in the form of KBP [LZ12].

We work on the game Hanabi, a collaborative card game where it’s natural to learn about the knowledge of other agents.
Epistemic logic

- Let $\mathcal{A}$, a set of agents
- Let $X$, a set of propositional atoms

Let $\mathcal{L}_{EL}$ the language := $\top | p | \neg \Phi | \Phi \lor \Phi | K_a \Phi$, $p \in X$, $a \in \mathcal{A}$

$K_i \phi$ means that "agent i knows that $\phi$".
Kripke structure

Kripke structure: \( \mathcal{M} = (W, R_1 \ldots R_n, V) \) [FHMV95]

- \( W \): set of worlds
- \( R_1 \ldots R_n \subseteq W \times W \), indistinguishability’s relations
- \( V \): valuation’s function \( W \rightarrow 2^X \)

Figure: Example of the knowledge of agents

Allows to interprets an epistemic formula in a certain state of knowledge.
Example in \( w_1 \): \( K_1p \), \( \neg K_2p \), \( K_1 \neg K_2p \) are true.
Event model

\[ \mathcal{E} = (E, E_1 \ldots E_n, pre, post) \]

- \( E \) : set of actions
- \( E_1 \ldots E_n \subseteq E \times E \) : indistinguishability's relations
- \( pre \) : precondition function, \( E \rightarrow \mathcal{L}_{EL} \)
- \( post \) : postcondition function, \( E \times X \rightarrow \mathcal{L}_{PROP} \)

**Figure**: Event model example
Product update

Product update : $\varepsilon$ et $\mathcal{M} : = \varepsilon \otimes \mathcal{M} = (W', R'_1 \ldots R'_n, V')$

- $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models \text{pre}(e)\}$
- $(w, e) R'_i (w', e')$ iff $w R_i w'$ and $e E_i e'$
- $V'((w, e)) = \{p \in X \mid \mathcal{M}, w \models \text{post}(e, p)\}$

It’s the cartesian product of the two Kripke structure.
Product update: example

Figure: \( \otimes \) Updated structure
KBP : Knowledge-based programs [LZ12]

A : set of primitive actions
KBP is defined inductively as follows :

1. the empty plan is a KBP.
2. any action $\alpha \in A$ is a KBP.
3. if $\pi$ and $\pi'$ are KBPs, then $\pi;\pi'$ is a KBP.
4. if $\Phi$ then $\pi$ else $\pi'$ is a KBP.
5. while $\Phi$ do $\pi$ is a KBP.

$\Phi$ must be subjective to the current agent.
Domain: We have set of V, set of actions

Problem: With an initial state and a goal (epistemic formula), we want to find KBP for each agent, such that when the agents execute their KBP synchronously turn by turn, the goal is reached in a finite number of steps.
Hanabi-like’s initial Kripke frame

Figure: Example of initial situation with 3 cards and 2 agents.

\( x_{ic} \) : agent \( i \) has card \( c \).
Event

A

j pioche 2

pre : xp2
post: xj2 ← ⊥
xp2 ← T

A

j

j pioche 1

pre : xp1
post: xj1 ← ⊥
xp1 ← T

A

j

j pioche 3

pre : xp3
post: xj3 ← ⊥
xp3 ← T
Toward an efficient representation

### Combinatorial explosion

Naive implementation of a classical graph: 2 players, 4 cards in hand.

<table>
<thead>
<tr>
<th>Cards</th>
<th>Number of worlds</th>
<th>Number of relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>630</td>
<td>22694</td>
</tr>
<tr>
<td>10</td>
<td>3150</td>
<td>58926</td>
</tr>
<tr>
<td>11</td>
<td>11550</td>
<td>112266</td>
</tr>
</tbody>
</table>

### Contribution

- Hanabi has a particularity: two worlds never have the same propositional valuations.

**Definition V-injective.** A Kripke structure $\mathcal{M} = \langle W, R_1 \ldots R_n, V \rangle$ is called *V-injective* if $V$ is injective, i.e.,

\[ \forall w, w' \in W: w \neq w' \implies V(w) \neq V(w'). \]

Also identified by M. Gattinger [Gat18].

Add variables to split worlds: [CS17].
**Definition** Boolean representation of Kripke Structure. Let $\mathcal{M}$ a $V$-injectif Kripke structure for agents $\mathcal{A} = \{a_1, \ldots, a_m\}$ and propositional variables $X = \{x_1, \ldots, x_n\}$. The propositional representation of $\mathcal{M}$ is a tuple of Boolean functions $F = \langle f_1, \ldots, f_m \rangle$ where every $f_i : \mathbb{B}^{2n} \rightarrow \mathbb{B}$ is defined as follow:

$$f_i(v_1, \ldots, v_n, v'_1, \ldots, v'_n) = 1 \iff \exists w, w' \in W: \begin{cases} \forall j: V(w)(x_j) = v_j \\ \forall j: V(w')(x_j) = v'_j \\ (w, w') \in R_i \end{cases}$$

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p'</th>
<th>q'</th>
<th>Rel</th>
<th>p</th>
<th>q</th>
<th>p'</th>
<th>q'</th>
<th>Rel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$w_1 \rightarrow w_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$w_1 \rightarrow w_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$w_2 \rightarrow w_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$w_2 \rightarrow w_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$w_1 \rightarrow w_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$w_2 \rightarrow w_1$</td>
</tr>
</tbody>
</table>

**Figure:** Representation by a Boolean function of the example
There is a practical algorithm for checking if a Boolean representation is a model of an epistemic formula.

Goal: find a propositional representation of $\Theta(F, \Phi)$ on $X$ of the set of worlds $Q(M, \Phi) = \{ w \in W \mid M, w \models \Phi \}$, where $\Phi \in \mathcal{L}_{EL}$.

i.e. $\text{Mod}(\Theta(F, \Phi)) = Q(M, \Phi)$.

**Proposition 1.** Let $F$ the propositional representation of $M$. Let $f_w$ is the formula which has for models all valuations of world in $W$.

1. $\Theta(F, x) = f_w \land x$
2. $\Theta(F, \neg \Phi) = f_w \land \neg \Theta(F, \Phi)$
3. $\Theta(F, \Phi \land \Psi) = \Theta(F, \Phi) \land \Theta(F, \Psi)$
4. $\Theta(F, \hat{K}_i \Phi) = \text{Forget}(f_i \land F', X'),$ where $F' = \Theta(F, \Phi)[X \rightarrow X']$

Several languages representing propositional formulas have efficient algorithms for these operations (OBDD for example).
Boolean representation for Event model

- Simple atoms are used for the original world: \( p \)
- Plus atoms are used to apply valuation in future state: \( p^+ \)
- With primes: modelise propositions of the arrival event

Here, \( \varphi_{e_1} = p \wedge \bar{p}^+ \) et \( \varphi_{e_2} = q \wedge \bar{q}^+ \)

We can obtain the event formula like this:

\[
\Phi_e = (\varphi_{e_1} \wedge \varphi_{e_1}[X' \leftarrow X, X^+ \leftarrow X^+]) \vee (\varphi_{e_1} \wedge \varphi_{e_2}[X' \leftarrow X, X^+ \leftarrow X^+]) \\
\vee (\varphi_{e_2} \wedge \varphi_{e_1}[X' \leftarrow X, X^+ \leftarrow X^+]) \vee (\varphi_{e_2} \wedge \varphi_{e_2}[X' \leftarrow X, X^+ \leftarrow X^+])
\]

<table>
<thead>
<tr>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Symbolic product update for propositional representation

**Proposition Product update.** with $f_i$ knowledge structure of agent $i$ and $\Phi_e$ the event formula:

$\text{Forget}(f_i \land \Phi_e, X \cup X')[X+ \leftarrow X, X'+ \leftarrow X']$
Here, we can modelize the knowledge of the agents. Now, we want to create KBP. We use regression because formulas include policy...
Regression

Starting from an epistemic goal formula, we want to get all the epistemic formulas that could lead to this goal formula through the events of the game.

**Definition Regression.** of $\Phi_G$ (goal formula) by $(M, w)$ (pointed event), called $\text{Reg}_w(\Phi_G)$ is the formula defined as (see [Auc12]):

\[
\begin{align*}
\text{Reg}_w(p) &= \text{Pre}(w) \land \text{Post}(w)(p) \\
\text{Reg}_w(\Phi \lor \Psi) &= \text{Reg}_w(\Phi) \lor \text{Reg}_w(\Psi) \\
\text{Reg}_w(\lnot \Phi) &= \text{Pre}(w) \land \lnot \text{Reg}_w(\Phi) \\
\text{Reg}_w(\hat{K}_j \Phi) &= \text{Pre}(w) \land \bigvee_{v' \in K_j(w)} \hat{K}_j(\text{Reg}_w(\Phi))
\end{align*}
\]
Data: n : degree regression
Data: final state
Result: Plan

\[ \Phi_G \leftarrow \text{final\_state}; \]
\[ \text{Plan} \leftarrow \text{list}((\Phi_G, 'stop')); \]
\[ \text{forall } i \in \{0 \ldots i\} \text{ do} \]
\[ \quad \text{tmp} \leftarrow \top; \]
\[ \quad \text{forall } a \in \text{Actions} \text{ do} \]
\[ \quad \quad \Phi_P \leftarrow \text{Reg}_a(\Phi_G); \]
\[ \quad \quad \text{Plan}.\text{append}((\Phi_P, a)); \]
\[ \quad \quad \text{tmp} \lor = \Phi_P; \]
\[ \quad \text{end} \]
\[ \quad \Phi_G \leftarrow \text{tmp}; \]
\[ \text{end} \]

Algorithm 1: Create plan

We obtain plan:

if \( \Phi_1 \) then execute action 1
else if \( \Phi_2 \) then execute action 2
else if \( \Phi_3 \) then execute action 3
... 
elise if \( \Phi_G \) then 'STOP'
Data: Plan

\(\text{done} \leftarrow \bot;\)

\textbf{while not done do}

\hspace{1em} \textbf{forall} (\(\Phi, \text{action}\) \(\in\) Plan \textbf{do}

\hspace{2em} \textbf{if} evaluate(\(\Phi, \text{state}\)) \textbf{then}

\hspace{3em} \textbf{if} \ action == \text{stop} \textbf{then}

\hspace{4em} \text{done} \leftarrow \top;

\hspace{3em} \text{end}

\hspace{3em} \text{Execute action ;}

\hspace{2em} \text{end}

\hspace{1em} \text{end}

\textbf{end}

\textbf{Algorithm 2:} Follow plan

Programm pointer of other agents in this KBP ?
Implementation in python with cudd library for BDD.
### Example

<table>
<thead>
<tr>
<th></th>
<th>A₁</th>
<th>A₂</th>
<th>Deck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id:</td>
<td>1 2</td>
<td>3 4</td>
<td>5 6 7 8 9 10</td>
</tr>
<tr>
<td>Value:</td>
<td>1 1</td>
<td>1 2</td>
<td>2 3 3 4 4 5</td>
</tr>
</tbody>
</table>

#### Regression Commune : Chaque agent a le même plan

<table>
<thead>
<tr>
<th>Etape1</th>
<th>Etape2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose(2, value 1 of A₂)</td>
<td>Choose(2, value 1 of A₁)</td>
</tr>
<tr>
<td>A₁ play 1</td>
<td>A₁ play 2</td>
</tr>
<tr>
<td>Choose(1, value 1 of A₂)</td>
<td>Choose(1, value 1 of A₁)</td>
</tr>
<tr>
<td>A₂ play 3</td>
<td>A₂ play 3</td>
</tr>
</tbody>
</table>

#### Regression Distribuée

<table>
<thead>
<tr>
<th>Etape1</th>
<th>Etape2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose(2, value 1 of A₂)</td>
<td>Choose(2, value 1 of A₁)</td>
</tr>
<tr>
<td>A₁ play 1</td>
<td>A₁ play 2</td>
</tr>
<tr>
<td>Choose(1, value 1 of A₂)</td>
<td>Choose(1, value 1 of A₁)</td>
</tr>
<tr>
<td>A₂ play 3</td>
<td>A₂ play 3</td>
</tr>
</tbody>
</table>
Conclusion, perspectives

- How to model Kripke structure with Boolean formula
- Quering this structure
- "Planing" with regression, but it’s too long.
  - eliminate redundant sub-formulas: requires an efficient data structure for epistemic formulas
  - $\Rightarrow$ Test with implementation of the Tableaux method, but formulas explode
- Exploit the initial state for the regression?
- Which heuristic for forward planification?
Guillaume Aucher.
Del-sequents for regression and epistemic planning.

Tristan Charrier and François Schwarzentruber.
A succinct language for dynamic epistemic logic.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi.
*Reasoning About Knowledge.*

Malvin Gattinger.
*New Directions in Model Checking Dynamic Epistemic Logic.*

Jérôme Lang and Bruno Zanuttini.
Knowledge-Based Programs as Plans - The Complexity of Plan Verification -.
Example OBDD

\[ f(x_1, \ldots, x_8) = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_7 x_8 \]
Atome $x_{pc}$ signifie le joueur $p$ possède la carte $c$.

$$\text{parmi}(0, X) = \bigwedge_{x \in X} \neg x$$

$$\text{parmi}(n, X) = \bigvee_{x \in X} x \land \bigwedge_{x \in X} (x \rightarrow \text{parmi}(n-1, X \setminus \{x\})),$$

avec $X$ une liste de variables propositionnelles quelconques.

**Unicité et existence d’une carte**

$$\mathcal{U} = \bigwedge_{c \in C} \text{parmi}(1, \text{vars}_c)$$

**Reciprocité**

$$\mathcal{R}_j = \bigwedge_{p \in P \setminus \{p,j\}} (\bigwedge_{c \in C} K_j x_{pc} \lor K_j \neg x_{pc})$$

**Denombrer**

$$\mathcal{D}_j = K_j \text{parmi}(n, \text{vars}_j)$$

avec $n =$ nombre de cartes maximal en main pour un joueur.