Towards an efficient representation for epistemic planning

Supervised by Alexandre Niveau

Sebastien Gamblin

University of Caen

December 6, 2018

Plan

Context of the work

- 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 langage $:= \top | \ p \mid \neg\Phi \mid \Phi \ \vee \ \Phi \mid K_a\Phi \ , \ \ p \in X \ , \ \ a \in \mathcal{A}$

$K_i\phi$ means that "agent i knows that ϕ".

Kripke structure

Kripke structure : $\mathcal{M} = (W, \mathcal{R}_1 \dots \mathcal{R}_n, V)$ [FHMV95]

- W : set of worlds
- $\mathcal{R}_1 \dots \mathcal{R}_n \subseteq W \times W$, indistinguishability's relations
- V : valuation's function $W \to 2^X$

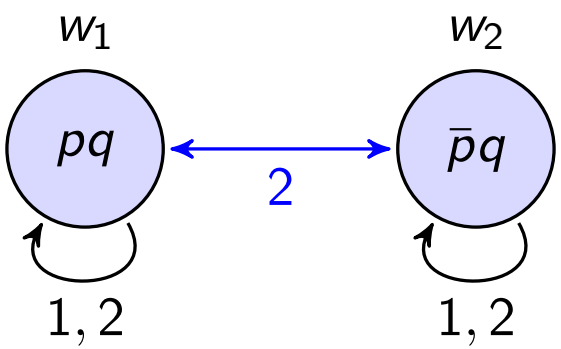


Figure: Example of the knowledge of agents

Allows to interprets an epistemic formula in a certain state of knowledge.
Example in w_1 : $K_1 p$, $\neg K_2 p$, $K_1 \neg K_2 p$ are true.

Event model

$\varepsilon = (E, \mathcal{E}_1 \dots \mathcal{E}_n, pre, post)$

- E : set of actions
- $\mathcal{E}_1 \dots \mathcal{E}_n \subseteq E \times E$: indistinguishability's relations
- pre : precondition function, $E \to \mathcal{L}_{EL}$
- $post$: postcondition function, $E \times X \to \mathcal{L}_{PROP}$

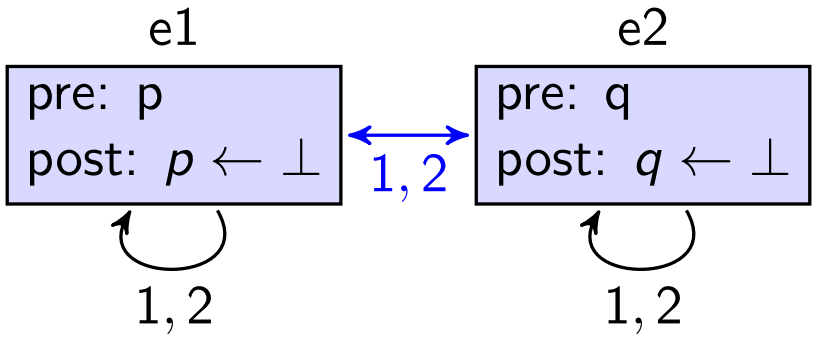


Figure: Event model example

Product update

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

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

It's the cartesian product of the two Kripke structure.

Product update: example

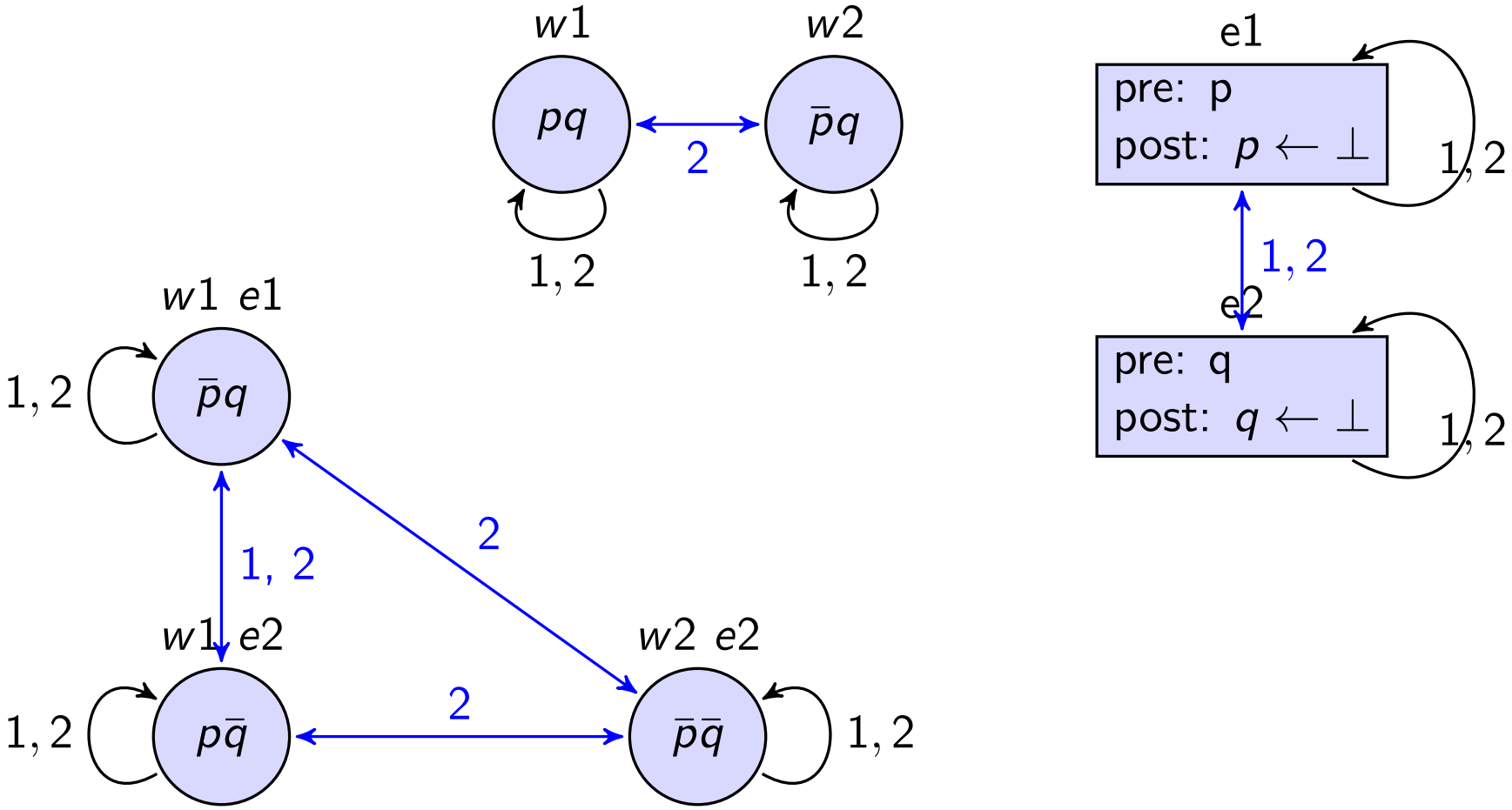


Figure: ⊗ Updated structure

KBP : Knowledge-based programs [LZ12]

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

- the empty plan is a KBP.
- any action $\alpha \in A$ is a KBP.
- if π and π' are KBPs, then $\pi; \pi'$ is a KBP.
- **if** Φ **then** π **else** π' is a KBP.
- **while** Φ **do** π is a KBP.

Φ 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

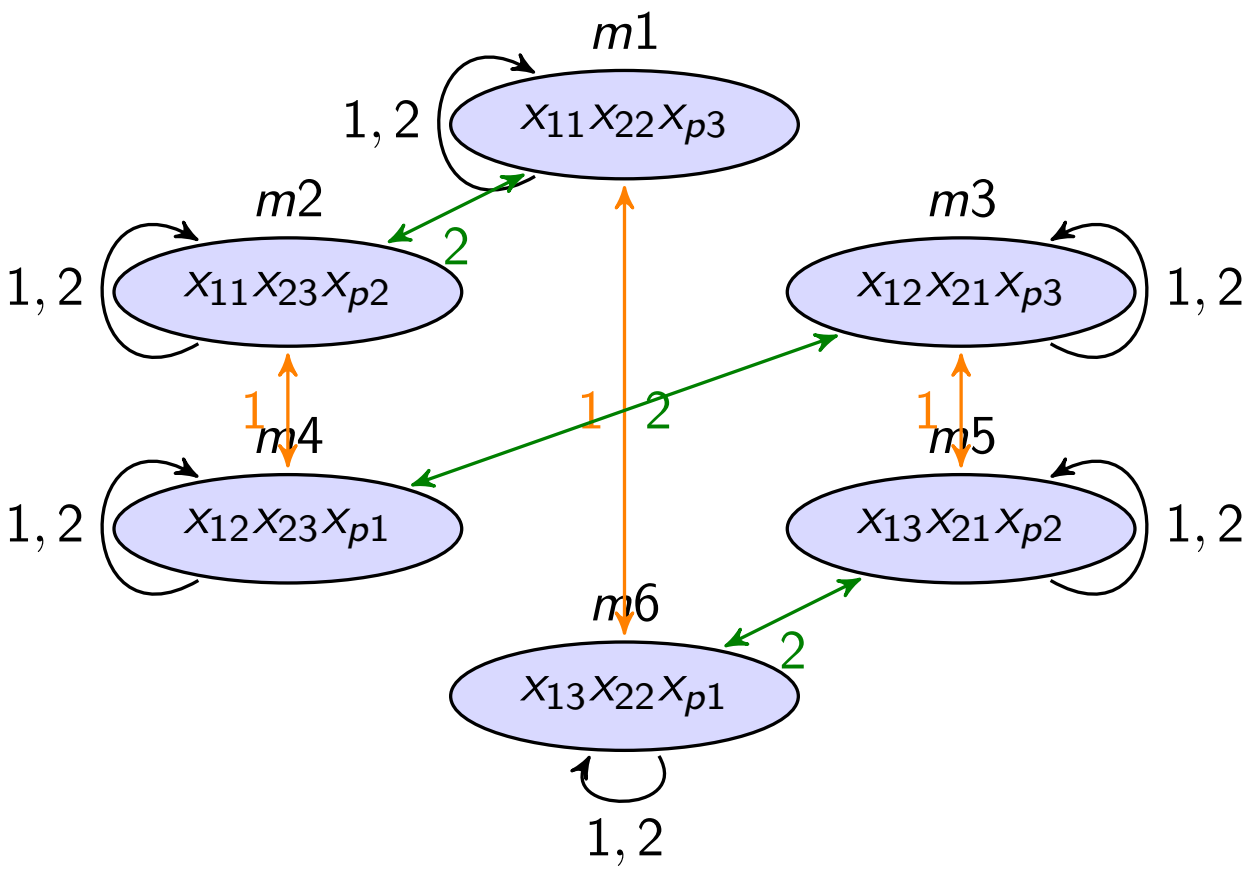
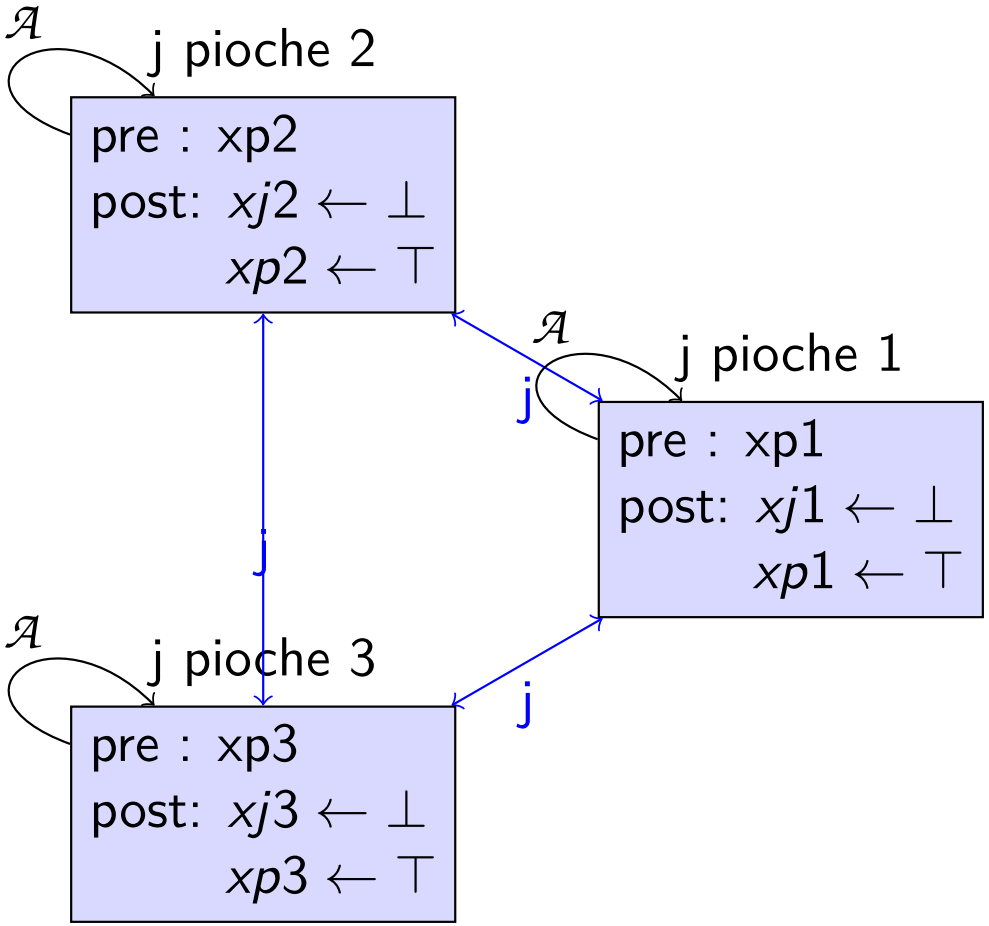


Figure: Example of initial situation with 3 cards and 2 agents.

x_{ic} : agent i has card c.

Event


$\mathcal{A}$
j pioche 2
pre : xp2
post: $xj2 \leftarrow \bot$
$xp2 \leftarrow \top$
$\mathcal{A}$
j pioche 1
pre : xp1
post: $xj1 \leftarrow \bot$
$xp1 \leftarrow \top$
$\mathcal{A}$
j pioche 3
pre : xp3
post: $xj3 \leftarrow \bot$
$xp3 \leftarrow \top$
j
j
j

Toward an efficient representation

Combinatorial explosion

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

Cards	Number of worlds	Number of relations
9	630	22694
10	3150	58926
11	11550	112266

Contribution

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

Definition V-injective. A Kripke structure $\mathcal{M} = \langle W, \mathcal{R}_1 \dots \mathcal{R}_n, V \rangle$ 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 propositonal 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 $f_i \colon \mathbb{B}^{2n} \to \mathbb{B}$ is defined as follow :

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

Relations for agent 1				
p	q	p'	q'	Rel
1	1	1	1	$w_1 \to w1$
0	1	0	1	$w_2 \to w_2$

Relations for agent 2				
p	q	p'	q'	Rel
1	1	1	1	$w_1 \to w1$
0	1	0	1	$w_2 \to w_2$
1	1	0	1	$w_1 \to w_2$
0	1	1	1	$w_2 \to w_1$

Figure: Representation by a Boolean function of the example

Model checking

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 $\mathcal{Q}(\mathcal{M}, \Phi) = \{w \in W \mid \mathcal{M}, w \models \Phi\}$, where $\Phi \in \mathcal{L}_{EL}$.
i.e. $Mod(\Theta(F, \Phi)) = \mathcal{Q}(\mathcal{M}, \Phi)$.

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

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

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 furtur 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 :

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

Relations								
p	$p+$	q	$q+'$	p'	$p'+$	q'	$q'+$	Rel
1	0			1	0			$e_1 \rightarrow e_1$
		1	0			1	0	$e_2 \rightarrow e_2$
1	0					1	0	$e_1 \rightarrow e_2$
		1	0	1	0			$e_2 \rightarrow e_1$

Symbolic product update for propositional representation

Proposition Product update. with f_i knowledge structure of agent i and Φ_e the event formula :
$Forget(f_i \wedge \Phi_e,\ X \cup X')[X+ \leftarrow X,\ X+' \leftarrow X']$

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 Φ_G (goal formula) by $(M,\ w)$ (pointed event), called $Reg_w(\Phi_G)$ is the formula defined as (see [Auc12]):

$$\begin{array}{lcl} Reg_w(p) & = & Pre(w) \wedge Post(w)(p) \\ Reg_w(\Phi \vee \Psi) & = & Reg_w(\Phi) \vee Reg_w(\Psi) \\ Reg_w(\neg\Phi) & = & Pre(w) \wedge \neg Reg_w(\Phi) \\ Reg_w(\widehat{K}_j\Phi) & = & Pre(w) \wedge \bigvee\limits_{v' \in K_j(w)} \widehat{K}_j(Reg_w(\Phi)) \end{array}$$

Data: n : degree regression
Data: final state
Result: Plan
$\Phi_G \leftarrow final\_state$;
$Plan \leftarrow list((\Phi_G, ' stop'))$;
forall $i \in \{0 \dots i\}$ **do**
  $tmp \leftarrow \top$;
  forall $a \in Actions$ **do**
    $\Phi_P \leftarrow Reg_a(\Phi_G)$;
    Plan.append$((\Phi_P, a))$;
    $tmp \vee = \Phi_P$;
  end
  $\Phi_G \leftarrow tmp$;
end

Algorithm 1: Create plan

We obtain plan :

if Φ_1 then execute action 1
else if Φ_2 then execute action 2
else if Φ_3 then execute action 3
...
elise if Φ_G then 'STOP'

```
Data: Plan
done ← ⊥;
while not done do
    forall (Φ, action) ∈ Plan do
        if evaluate(Φ, state) then
            if action == stop then
                done ← ⊤ ;
            end
            Execute action ;
        end
    end
end
```

Algorithm 2: Follow plan

Programm pointer of other agents in this KBP ?

Implementation in python with cudd library for BDD.

Example

	A_1		A_2		Deck					
Id :	1	2	3	4	5	6	7	8	9	10
Value :	1	1	1	2	2	3	3	4	4	5

	Regression Commune : Chaque agent a le même plan			
	xt_1	xt_2	xt_3	xt_1 ou xt_2 ou xt_3
Planning time	494	478	500	1458
Execution time	224	217	117	285
Total	737	712	635	1760
Etape1	Choose(2, value 1 of A_2)	Choose(2, value 1 of A_1)	Choose(1, value 1 of A_2)	Choose(1, value 1 of A_2)
Etape2	A_1 play 1	A_1 play 2	A_2 play 3	A_2 play 3
	Regression Distribuée			
	xt_1	xt_2	xt_3	xt_1 ou xt_2 ou xt_3
Planning time	68	68	70	216
Execution time	77	72	162	182
Total	162	156	249	418
Etape1	Choose(2, value 1 of A_2)	Choose(2, value 1 of A_1)	Choose(1, value 1 of A_2)	Choose(1, value 1 of A_2)
Etape2	A_1 play 1	A_1 play 2	A_2 play 3	A_2 play 3

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
  - -> 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 plannning.
Journal of Applied Non Classical Logics, page 29, 2012.

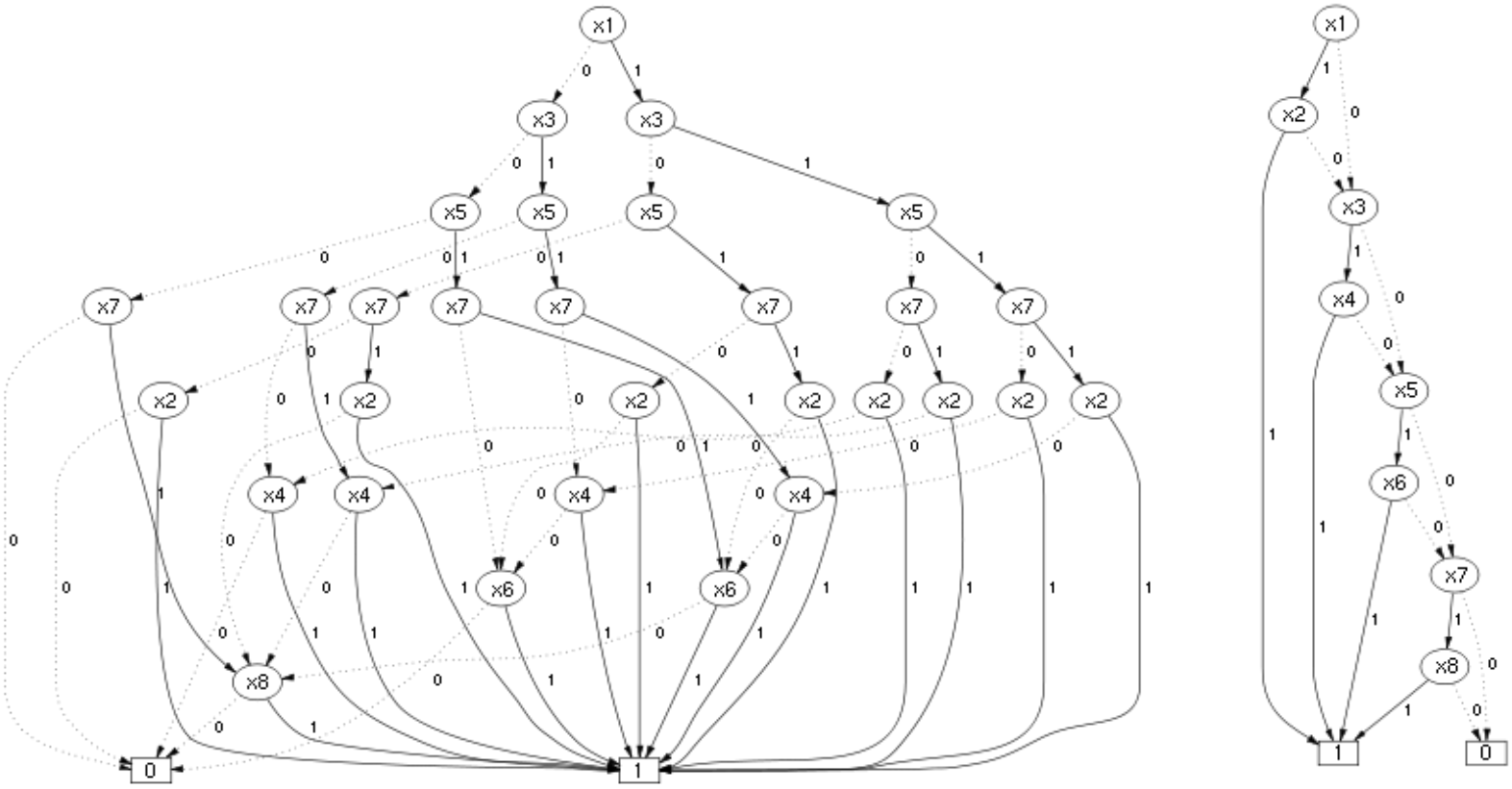
Tristan Charrier and Francois Schwarzentruber.
A succinct language for dynamic epistemic logic.
In *Proc. 16th Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2017)*, pages 123–131, 2017.

Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi.
Reasoning About Knowledge.
The MIT PRess, Cambdrige, Massachusetts, 1995.

Malvin Gattinger.
New Directions in Model Checking Dynamic Epistemic Logic.
PhD thesis, Universiteit van Amsterdam, 2018.

Jérôme Lang and Bruno Zanuttini.
Knowledge-Based Programs as Plans - The Complexity of Plan Verification -.
In *Proc. 20th Conference on European Conference on Artificial Intelligence (ECAI 2012)*, page 6 p., France, August 2012.

Example OBDD



$$f(x_1, \ldots, x_8) = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_7 x_8$$

Modelisation of Hanabi

Atome $x_p c$ signifie le joueur p possede la carte c.

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

parmi(n, X) $= \bigvee_{x \in X} x \wedge \bigwedge_{x \in X} (x \rightarrow$ parmi(n-1, $X \backslash \{x\}$)),

avec X une liste de variables propositionnelles quelconques.

Unicite et existence d'une carte $\mathcal{U} = \bigwedge_{c \in \mathcal{C}} parmi(1,\ vars_c)$

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

Denombrer $\mathcal{D}_j = K_j\ parmi(n,\ vars_j)$

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