Knowledge-based programs as plans

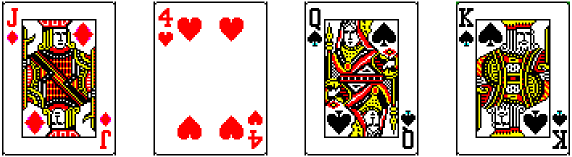
Jérôme Lang (LAMSADE, Paris) & Bruno Zanuttini (GREYC, Caen)

ECAI-2012 + TARK-2013 + IJCAI-2015
+ ongoing work (with Anaëlle Wilczinski, LAMSADE)

A card game program

Goal :

- pick some cards, maximum 5
- try to obtain three cards of the same rank



```
Do
    pick a card $c$
    look at the rank of $c$
Until three cards of the same rank or know it is impossible
```

A diagnose-and-repair program

- three components 1,2,3 ;
- propositional symbol ok_i : component i is in working order ;
- action $repair(i)$: makes ok_i true ;
- action $test(i)$: returns the truth value of ok_i ;
- initial knowledge state : $K((ok_1 \leftrightarrow (ok_2 \wedge ok_3)) \wedge (\neg ok_1 \vee \neg ok_3))$;
- Goal : to have the three components working without replacing more components than necessary.

```
While $\neg K(ok_1 \wedge ok_2 \wedge ok_3)$ do
    $i$ := smallest integer such that $\neg K ok_i$ ;
    If $\neg K \neg ok_i$ then $test(i)$ endif ;
    If $K \neg ok_i$ then $replace(i)$ endif
Endwhile
```

Outline

Knowledge-based programs :

- introduced by Fagin, Halpern, Moses and Vardi [1995]
- studied for behaviour specification in distributed environments
- we use them as *outputs of planning problems*
- *what are the benefits and pitfalls of using knowledge-based programs instead of standard programs ?*

Classical partially observable planning vs. Knowledge-based planning

Classical partially observable planning

Output = standard plan (policy) :

- tree or DAG containing observations/actions
- branching on current state and observations

Knowledge-based planning

Output = knowledge-based program :

- branching conditions are subjective epistemic formulas

Example

- initial knowledge state : $O((ok_1 \leftrightarrow (ok_2 \wedge ok_3)) \wedge (\neg ok_1 \vee \neg ok_3))$
- goal knowledge state : $K(ok_1 \wedge ok_2 \wedge ok_3)$
- actions : test(i), repair(i) for $i = 1, 2, 3$

Knowledge-based plan :

```
**While** $\neg K(ok_1 \wedge ok_2 \wedge ok_3)$ **do**
    find the smallest $i$ such that $\neg K ok_i$ ;
    **If** $\neg K \neg ok_i$ **then** $test(i)$ ;
    **If** $K \neg ok_i$ **then** $replace(i)$
**Endwhile**
```

Knowledge-based plans vs. policies

KBP

```
While $\neg K(ok_1 \wedge ok_2 \wedge ok_3)$ do
    find smallest $i$ such that $\neg K ok_i$ ;
    If $\neg K \neg ok_i$ then $test(i)$ ;
    If $K \neg ok_i$ then $replace(i)$
Endwhile
```

standard policy

```
$replace(1)$ ;
$test(2)$ ;
If      $ok(2)$
then    $replace(3)$
else    $replace(2)$ ;
        $test(3)$ ;
        If $\neg ok(3)$
        then $replace(3)$
        endif
endif
```

Knowledge-based programs vs. standard programs

Knowledge-based programs :

- introduced by Fagin, Halpern, Moses and Vardi [1995]
- studied for behaviour specification in distributed environments
- we use them as *outputs of planning problems*
- *what are the benefits and pitfalls of using knowledge-based programs instead of standard programs ?*
- [-] more difficult to execute than standard programs : evaluating branching conditions is computationally hard
- [+] more compact than standard programs
- [+] more natural to express than standard programs

Outline

Knowledge-based programs :

- introduced by Fagin, Halpern, Moses and Vardi [1995]
- studied for behaviour specification in distributed environments
- we use them as *outputs of planning problems*.

Our work :

- using knowledge-based programs as (single-agent) plans reaching some goals described by epistemic formulas
- LOFT-12 / ECAI-12 : expressivity and complexity of plan verification
- TARK-13 : comparing the succinctness of KBPs to that of standard plans + complexity of plan existence
- IJCAI-15 : probabilistic knowledge-based programs
- ongoing work : KBP synthesis
- ongoing work : multi-agent KBP

Plan

Knowledge-based programs

Knowledge-based planning problems

Succinctness

KBP verification

KBP existence

Probabilistic KBPs

KBP synthesis

KBP synthesis

Multi-agent KBPs

Knowledge-based programs

Knowledge-based planning problems

Succinctness

KBP verification

KBP existence

Probabilistic KBPs

KBP synthesis

KBP synthesis

Multi-agent KBPs

Syntax

Input

- set of propositional variables $X = \{x_1, \ldots, x_n\}$
  - Queen(c_1), ok_1 ...
  - state = truth assignment (unobservable)
- set of actions

Knowledge-based program π :

- action, or
- sequence $\pi_1; \pi_2; \ldots; \pi_n$, or
- branching If Φ then π_1 else π_2, where Φ is a purely subjective S5 formula (Boolean combination of epistemic atoms $K\varphi$) ; or
- loop While Φ do π_1, where Φ is a purely subjective S5 formula.

Actions

Ontic action :

- changes the state of the world
- possibly nondeterministic + no feedback
- propositional symbol $x \mapsto \{x, x'\}$;
  - x before the action is performed
  - x' after the action is performed
- switch(x_i) : $\Sigma = (x'_i \leftrightarrow \neg x_i) \wedge \bigwedge_{j \neq i} (x'_j \leftrightarrow x_j)$
- $x_i \leftarrow 0$: $\Sigma = (\neg x'_i)$
- reinit(x_i) : $\Sigma = \bigwedge_{j \neq i} (x'_j \leftrightarrow x_j)$

Epistemic action :

- does not change the state of the world
- sends back one of several possible observations
- test($x_i \vee x_j$) : observe $x_i \vee x_j$ or observe $\neg(x_i \vee x_j)$
- ask-how-much-time-left : observe $(t = 15mn)$ or observe $(t = 10mn)$ or observe $(t = 5mn)$ or observe $(t = 0)$

Executing a KBP

At every step :

- current state of variables s^t
  - $s^0 = x_1 x_2 \bar{x}_3$
- current knowledge state M^t
  - $M^t = \{x_1 x_2 x_3, x_1 \bar{x}_2 x_3, x_1 x_2 \bar{x}_3\}$
  - succinct representation $O(x_1 \wedge (x_2 \vee x_3))$: *all I know* is $x_1 \wedge (x_2 \vee x_3)$.

Execution :

- branching condition / loop : evaluated in M^t
- ontic action : nondeterministic modification of s^t
- epistemic action :
  - no modification of s^t
  - reception of an observation ω

Progression

Progression by an ontic action :

- $M^t = \{x_1 x_2 x_3, \bar{x}_1 \bar{x}_2 \bar{x}_3\}$ $O((x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge \neg x_3))$
- progression of M^t by switch(x_1) :
  $M^{t+1} = \{\bar{x}_1 x_2 x_3, x_1 \bar{x}_2 \bar{x}_3\}$ $O((\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3))$
- progression of M^{t+1} by reinit(x_1) :
  $M^{t+2} = \{x_1 x_2 x_3, \bar{x}_1 x_2 x_3, x_1 \bar{x}_2 \bar{x}_3, \bar{x}_1 \bar{x}_2 \bar{x}_3\}$ $O(x_2 \leftrightarrow x_3)$

Progression by an observation (received after some epistemic action) :

- action test($x_1 \wedge x_2$), observation $\neg(x_1 \wedge x_2)$:
- progression of M^{t+2} by observation $\neg(x_1 \wedge x_2)$:
  $M^{t+3} = \{\bar{x}_1 x_2 x_3, x_1 \bar{x}_2 \bar{x}_3, \bar{x}_1 \bar{x}_2 \bar{x}_3\}$ $O((x_2 \leftrightarrow x_3) \wedge \neg(x_1 \wedge x_2))$

Classical planning

- Set of initial states and goal states (described succinctly)
- Set of actions whose effects are described succinctly
- Output : standard plan (policy) :
  - tree or DAG containing observations/actions
  - branching on current state and observations

Knowledge-based planning problems

- initial knowledge state initial M^0 :
  - possibly $O\top$
  - must contain the true initial state
- goal G (purely subjective epistemic formula)
- π valid plan if
  - terminates
  - for every possible sequence of states $s^0 \in M^0 \dots s^{\text{final}} \in M^{\text{final}}$ we have $s^{\text{final}} \models G$

Example

- initial knowledge state : $O((ok_1 \leftrightarrow (ok_2 \wedge ok_3)) \wedge (\neg ok_1 \vee \neg ok_3))$
- goal knowledge state : $K(ok_1 \wedge ok_2 \wedge ok_3)$
- actions : test(i), repair(i) for $i = 1, 2, 3$

Knowledge-based plan :

```
While $\neg K(ok_1 \wedge ok_2 \wedge ok_3)$ do
    find the smallest $i$ such that $\neg K ok_i$ ;
    If $\neg K \neg ok_i$ then $test(i)$ ;
    If $K \neg ok_i$ then $replace(i)$
Endwhile
```

Knowledge-based plans vs. standard policies

- A standard policy is a KBP in which the last action executed before any branching condition *if* Φ or *while* Φ is an epistemic action a such that Φ is one of the possible observations for a.
- For every KBP π there exists a standard policy π' "equivalent" to π (π and π' have the same execution traces).

Expressivity :

- there exists a valid knowledge-based for a planning problem P iff there exists a valid standard policy for P

Knowledge-based plans vs. policies

KBP

```
While $\neg K(ok_1 \wedge ok_2 \wedge ok_3)$ do
    find smallest $i$ such that $\neg K ok_i$ ;
    If $\neg K \neg ok_i$ then $test(i)$ ;
    If $K \neg ok_i$ then $replace(i)$
Endwhile
```

standard policy

```
$replace(1)$ ;
$test(2)$ ;
If      $ok(2)$
then    $replace(3)$
else    $replace(2)$ ;
        $test(3)$ ;
        If $\neg ok(3)$
        then $replace(3)$
        endif
endif
```

Knowledge-based plans vs. policies : reactivity

On-line execution :

- standard policy :
  - move to the subtree corresponding to the observation and execute the next action
  - constant time
- knowledge-based plan :
  - branching / loop condition : decide $M^t \models \Phi$
  - NP-hard and coNP-hard, in $\Delta_2 P$

Knowledge-based plans vs. policies : succinctness

Proposition : unless NP $\subseteq$ P/poly (extremely unlikely), while-free KBPs with atomic branching conditions are exponentially more succinct than while-free standard policies.

Proof sketch :

- for each $n \in N$ we build a polysize KBP π_n that "reads" a CNF formula φ and either makes sure that it is unsatisfiable or else builds a model of it.
- if there is a family of standard policies π'_n for every n, of size polynomial in $|\pi_n|$, with π_n equivalent to π'_n, then there is a (possibly nonuniform) polytime algorithm for $3\textsc{sat}$, yielding NP $\subseteq$ P/poly.

Knowledge-based plans vs. policies : succinctness

Proposition : KBPs (with loops) are more succinct than standard policies (with loops).

Proof sketch :

- there is a polynomial *pol* and a collection of KBPs $(\pi_n)_n$ such that $|\pi_n| \leq pol(n)$ and such that π_n "counts" up to $2^{2^n} - 1$ (by going once through all knowledge states).
- we build a family of planning problems $(P_n)_n$ such that the only valid plans for P_n are all equivalent to π_n
- assume that for all n there is a standard policy π'_n for P_n and $|\pi'_n| \leq pol(n)|$; then π'_n can manipulate only $pol(n)$ variables, and can have only $2^{pol(n)}.|\pi'_n|$ configurations (states + control points) ; then it cannot count up to $2^{2^n} - 1$, contradiction.

Knowledge-based plans vs. policies : succinctness

Proposition : KBPs are more succinct than while-free KBPs.

Proof sketch : later

Knowledge-based programs

Knowledge-based planning problems

Succinctness

KBP verification

KBP existence

Probabilistic KBPs

KBP synthesis

KBP synthesis

Multi-agent KBPs

KBP existence vs. KBP verification

KBP verification

input $P =$ (initial belief state, actions, goal)
question is π valid for P ?

KBP existence

input $P =$ (initial belief state, actions, goal)
question is there a valid KBP π for P ?

small KBP existence

input $P = +$ integer k encoded in unary
question is there a valid KBP π for P such that $|\pi| \leq k$?

KBP verification : overview of results

loop-free programs

- Π_2^P-complete ($\Pi_2^P = \text{coNP}^{\text{NP}}$)
- remains Π_2^P-complete with each of the following restrictions :
  - ontic actions only
  - epistemic actions only
- standard plan verification : coNP-complete

programs with loops

- EXPSPACE-complete
- remains EXPSPACE-complete even if we know that π terminates
- standard plan verification : PSPACE-complete

KBP verification : loop-free programs

- Π_2^P-complete ($\Pi_2^P = \mathrm{coNP}^{\mathrm{NP}}$)
- hardness proof easy
- membership proof based on the following nondeterministic algorithm that shows that a plan is *not* valid ;
    - guess a sequence of observations
    - at each step with a branching condition Φ, evaluate Φ [Needs a polynomial number of NP oracles]
    - check that the goal is not satisfied at the end of the execution

KBP verification : general programs

- EXPSPACE-complete
- key point : a loop can be executed up to $2^{2^n} - 2$ times (visit all possible belief states)
- membership easy
- hardness by reduction from NONDETERMINISTIC UNOBSERVABLE PLAN EXISTENCE (Haslum and Jonsson, 99)

Knowledge-based plans vs. policies : succinctness

Proposition : KBPs are more succinct than while-free KBPs.

Proof sketch :

- verifying a KBP with loops is EXPSPACE-complete ;
- verifying a while-free KBP is Π_2^p-complete ;
- $\Pi_2^p \subseteq$ PSPACE $\subset$ EXPSPACE (strict inclusion, Savitch's theorem)

KBB existence : overview of results

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

Corollaries from known results in planning together with the fact that there exists a KBP for a planning problem off there exists a standard plan.

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

- membership : guess π of size $\leq k$ and verify it ; PLAN VERIFICATION is in EXPSPACE and NEXPSPACE = EXPSPACE.
- hardness : reduction from PLAN VERIFICATION. Build a planning problem P', and let $k = |\pi|$, such that every valid plan for P' is equivalent to π.

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

- membership : guess π and verify it ; PLAN VERIFICATION is in Π_2^p.
- hardness : reduction from $\text{QBF}_{3,\exists}$.

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

- branching is not necessary because we never get any feedback ; the problem is equivalent to polynomially-bounded plan existence without branching, which is Σ_2^p-complete.

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

- membership : because an epistemic action needs to be executed at most once, if a planning problem has a valid KBP then it has a valid KBP of height bounded by the number of epistemic actions.
  $+$ searching a polynomial-height tree can be done in PSPACE.
- hardness : reduction from QBF.

KBP existence

	unbounded	bounded
general	2-EXPTIME-complete	EXPSPACE-complete
while-free	2-EXPTIME-complete	Σ_3^p-complete
ontic	EXPSPACE-complete	?
while-free, ontic	EXPSPACE-complete	Σ_2^p-complete
while-free, epistemic	PSPACE-complete	?
while-free, epistemic, positive goal	coNP-complete	Σ_2^p-complete

- membership, unbounded : performing an epistemic action cannot harm ; there exists a valid KBP iff the KBP consisting in performing all epistemic actions in any order is valid.
- membership, bounded : guess a set of k epistemic actions and perform them ; verification is in coNP.
- hardness : reductions from UNSAT and $\text{QBF}_{2,\exists}$.

Probabilistic Belief-Based Programs

- 5 doors :
  - a tiger hidden behind two of them
  - a princess behind one of the other three
  - initially, all possible configurations equiprobable.
- sensing actions $listen_i$, $i = 1, \ldots, 4$ (not 5). Feedback :
  - if a tiger is behind door i : hear the tiger roaring (r_+) with probability 0.5, or not (r_-) with probability 0.5
  - if no tiger behind door i : r_- with probability 1 ;
- ontic actions $open_i$: $i = 1, \ldots, 5$. Effects : the agent...
  - ... becomes eaten by the tiger if there is one behind door i (reward -1)
  - ... becomes married to the princess if she is behind door i (reward $+1$)

Probabilistic Belief-Based Programs

π :

```
$listen_1$; $listen_2$; $listen_3$; $listen_4$;
while $P(t_1) > 0.1 \wedge \cdots \wedge P(t_5) > 0.1$ do
    if $P(t_1) \leq P(t_2) \wedge \cdots \wedge P(t_1) \leq P(t_5)$ then
        [ $listen_1$; if $P(t_1) \leq 0.1$ then $open_1$ ]
    elseIf $P(t_2) \leq P(t_1) \wedge \cdots \wedge P(t_2) \leq P(t_5)$ then
        [ $listen_2$; if $P(t_2) \leq 0.1$ then $open_2$ ]
    ...
    else [ if $P(t_5) \leq 0.1$ then $open_5$ ]
```

π corresponds to a (less succinct) POMDP policy, with branching on sequences of observations.

KBP synthesis

Informally : maintain a list L of pairs $\langle K\varphi, a\rangle$ such that performing a in knowledge state $K\varphi$ eventually leads to the goal

- L initialized to $\{\langle K\varphi, \mathit{stop}\rangle \mid K\varphi \in \Gamma\}$
- repeat
    - $\Gamma' = \bigvee\{K\varphi \mid \langle K\varphi, a\rangle \in L \text{ for some } a\}$
    - regress Γ' by some action α
    - add $\langle \mathit{Reg}(\Gamma', \alpha), \alpha\rangle$ to L (unless it is redundant)
- until the initial knowledge state implies $K\varphi$ for some $\langle K\varphi, a\rangle$ in L

If $L = \{\langle\varphi_i, \alpha_i\rangle, i = 1, \ldots, m\}$, return

```
REPEAT
    CASE
        $\varphi_1 : \alpha_1$
        ...
        $\varphi_m : \alpha_m$
    END
UNTIL  stop
```

KBP synthesis

Same example as in (Herzig, Lang & Marquis, 2003) :

- two propositional variables u, v
- epistemic actions $\alpha = \mathit{test}(u \wedge v)$, $\beta = \mathit{test}(u \leftrightarrow v)$
- ontic action $\gamma = \mathit{switch}(u)$
- initial knowledge state $K\top$
- goal $Kv \vee K\neg v$.

Successive values of L :

1. initially : $L = \{\langle Kv, \mathit{stop}\rangle, \langle K\neg v, \mathit{stop}\rangle\}$
2. add $\langle K(v \rightarrow u), \alpha\rangle$
3. add $\langle K(v \rightarrow \neg u), \gamma\rangle$
4. add $\langle K\top, \beta\rangle\}$

$$L = \left\{ \ \langle Kv, \mathit{stop}\rangle, \langle K\neg v, \mathit{stop}\rangle, \langle K(v \rightarrow u), \alpha\rangle, \langle K(v \rightarrow \neg u), \gamma\rangle, \langle \top, \alpha\rangle \ \right\}$$

KBP synthesis

The plan returned is

```
REPEAT
    CASE
        $Kv$ :               $stop$
        $K\neg v$ :           $stop$
        $K(v \to u)$ :       $\alpha$
        $K(v \to \neg u)$ :  $\gamma$
        $K\top$ :            $\beta$
    END
UNTIL stop
```

Multi-agent KBPs : three prisoners and a lightbulb

The propositional variables :

- *in*(i) : i is in the room
- *hasbeen*(i) : i has already been in the room
- *light* : the light is switched on
- *success*
- *end* (ensures *tell* is performed successfully at most once)

Multi-agent KBPs : three prisoners and a lightbulb

The actions :

- *wait*(i) : nature possibly sends one of the agents into the room ; i learns whether he is in the room or not ; i can be sent in the room if it is not empty ; i forgets about the light if he knew something about it.

$$(K_i\ in(i) \vee K_i\ \neg in(i)) \wedge K_i \bigwedge_{j \neq k} (in(j)' \rightarrow (\neg in(k) \wedge \neg in(k)') \wedge (...))$$

- *observe*(i) : i learns the value of l, provided that he is in the room :

$$K_i(in(i) \rightarrow light) \vee K(in(i) \rightarrow \neg light) \wedge (...)$$

- *switch* : $K_i(l' \leftrightarrow \neg l) \wedge (...)$
- *exit*(i) : i exits the room if he was in it : $K_i \neg in(i) \wedge (...)$
- *tell* : $K_i\,(end' \wedge (hasbeen(1) \wedge hasbeen(2) \wedge \neg end \rightarrow success') \wedge (...))$

and all these actions theories are common knowledge

Multi-agent KBPs : three prisoners and a lightbulb

```
$\pi_0$ :

if $K_0\ \neg in(0)$ then
    wait(0)
else
    observe(0) ;
    if $K_0(hasbeen(1) \wedge hasbeen(2))$ then
        tell
    else
        if $K_0$ light then
            switch
        end if
    end if
    exit
end if
```

Multi-agent KBPs : three prisoners and a lightbulb

```
$\pi_1$ :

if $K_1$ $\neg$in(1) then
    wait(1)
else
    if $K_1\neg$hasbeen(1) then
        observe(1) ;
        if $K_1$ $\neg$light then
            switch
            hasbeen(1) := true
        end if
    end if
    exit
end if
```

π_2 is the same as π_2, replacing 1 by 2 everywhere.