Knowledge-Based Policies for Qualitative Decentralized POMDPs

Abdallah Saffidine Bruno Zanuttini
François Schwarzentruber

May 14th, 2019

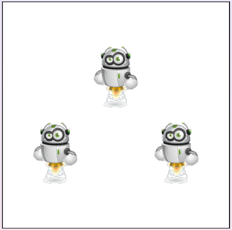
Automation of complex tasks

Building surveillance

Nuclear decommissioning

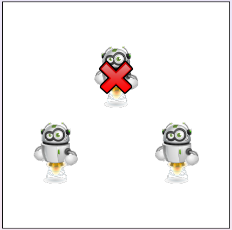
Intelligent farming

Multiple robots



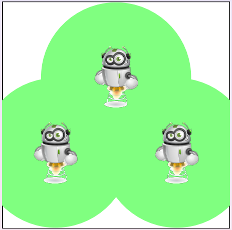
more robust/efficient than

Multiple robots



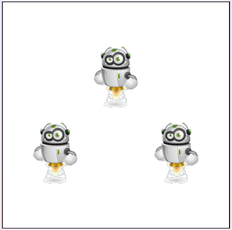
more **robust**/efficient than

Multiple robots



more robust/**efficient** than

Multiple robots

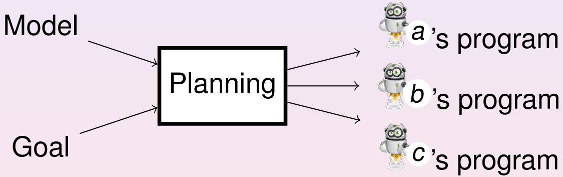


more robust/efficient than

Settings

- Cooperative agents;
- Common goal;
- Imperfect information;
- Decentralized execution.

Methodology



Model

Goal

Planning

a's program

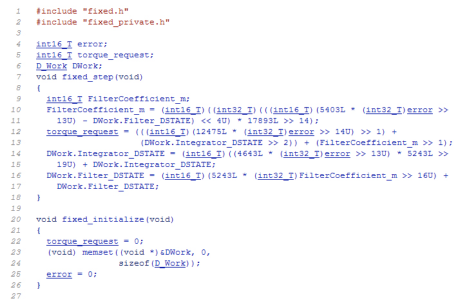
b's program

c's program

Need: understandable system

Motivation

- Legal issues in case of failure
- Interaction with humans



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

int16_T error;
int16_T torque_request;
D_Work DWork;
void fixed_step(void)
{
  int16_T FilterCoefficient_m;
  FilterCoefficient_m = (int16_T)((int32_T)(((int16_T)(5403L * (int32_T)error >>
    13U) - DWork.Filter_DSTATE) << 4U) * 17893L >> 14);
  torque_request = (((int16_T)(12475L * (int32_T)error >> 14U) >> 1) +
                    (DWork.Integrator_DSTATE >> 2)) + (FilterCoefficient_m >> 1);
  DWork.Integrator_DSTATE = (int16_T)((4643L * (int32_T)error >> 13U) * 5243L >>
    19U) + DWork.Integrator_DSTATE;
  DWork.Filter_DSTATE = (int16_T)(5243L * (int32_T)FilterCoefficient_m >> 16U) +
    DWork.Filter_DSTATE;
}

void fixed_initialize(void)
{
  torque_request = 0;
  (void) memset((void *)&DWork, 0,
                sizeof(D_Work));
  error = 0;
}

```

Our contribution: use of knowledge-based programs

KBP for agent *a*

```
listenRadio
if a knows strike
    toStation
else
    toAirport
```

KBP for agent *b*

```
readNewsPaper
if b knows strike
    toStation
else
    toAirport
```

✓

- Operational Semantics for Knowledge-based programs;
- Succinctness;
- (Un)decidability/complexity.

Extends: [Lang, Zanuttini, ECAI2012, TARK2013]

Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
4. Conclusion

Program constructions

Language constructions

turn left

stay

broadcast temperature

...; ...

if φ **then** ...**else** ...

while φ **do** ...

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

```
if a knows (door 12 is locked and justobserved(🔥)) then
        turn left
        broadcast temperature
else
        stay
```

Outline

1. Knowledge-based programs
2. Semantics
   - Models: QdecPOMDP
   - Operational semantics of KBPs
3. Mathematical Properties
4. Conclusion

Outline

1 Knowledge-based programs

2 Semantics
- Models: QdecPOMDP
- Operational semantics of KBPs

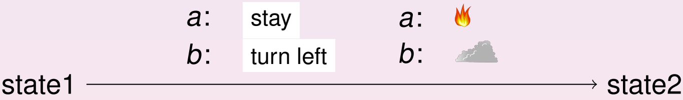
3 Mathematical Properties

4 Conclusion

QdecPOMDP

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

Transitions of the form:



a: stay *a*: 🔥

b: turn left *b*: ☁

state1 ⟶ state2

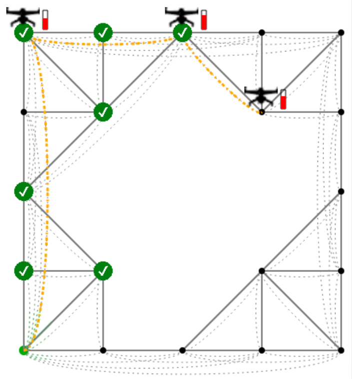
A non-empty set of possible initial states;

A set of goal states.

States

Typically, a state describes:

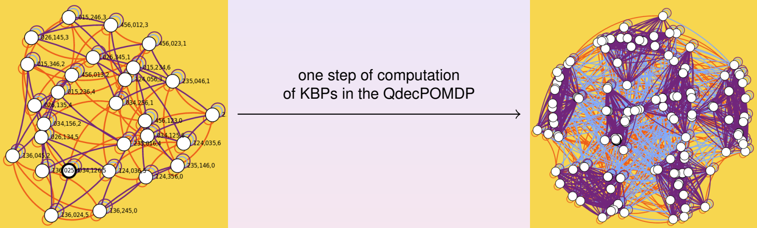
- positions of agents;
- battery levels;
- etc.



Outline

1. Knowledge-based programs
2. Semantics
    - Models: QdecPOMDP
    - Operational semantics of KBPs
3. Mathematical Properties
4. Conclusion

Operational semantics



one step of computation
of KBPs in the QdecPOMDP

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;
- synchrony of the system;
  - tests last 0 unit of time;
  - actions last 1 unit of time.

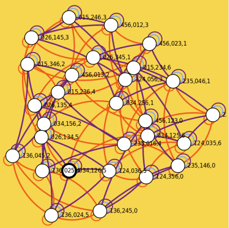
KBP for agent *a*

```
listenRadio
if a knows strike
    toStation
else
    toAirport
```

KBP for agent *b*

```
readNewsPaper
if b knows strike
    toStation
else
    toAirport
```

Epistemic structures at time T: worlds



Worlds = consistent histories of the form
(wait few slides)

$$s^0\overrightarrow{pc}^0 \boxed{\overrightarrow{obs}^1 s^1 \overrightarrow{pc}^1} \quad \ldots \quad \boxed{\overrightarrow{obs}^T s^T \overrightarrow{pc}^T}$$

where

$\overrightarrow{obs}^t$	vector of observations at time t
s^t	state at time t
$\overrightarrow{pc}^t$	vector of program counters at time t

```
● listenRadio
■if $K_a strike$ then
|        toStation
else
|        toAirport
▲
```

Epistemic structures at time t: indistinguishability relations

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

$$\begin{array}{c} s^0\overrightarrow{pc}^0\boxed{\overrightarrow{obs}^1 s^1\overrightarrow{pc}^1}\quad\dots\quad\boxed{\overrightarrow{obs}^T s^T\overrightarrow{pc}^T} \\ \sim_a \\ s'^0\overrightarrow{pc}'^0\boxed{\overrightarrow{obs}'^1 s'^1\overrightarrow{pc}'^1}\dots\boxed{\overrightarrow{obs}'^T s'^T\overrightarrow{pc}'^T}\end{array} \quad \text{iff} \quad \begin{array}{c}\text{for all } t\in\{1,\dots,T\},\\ \overrightarrow{obs}^t_a = \overrightarrow{obs}'^t_a\end{array}$$

Program counters

Definition (Program counter)

(guard, action just executed, continuation)

```
● listenRadio
■if $K_a strike$ then
    toStation
else
    toAirport
▲
```

($\top$, start, ●)

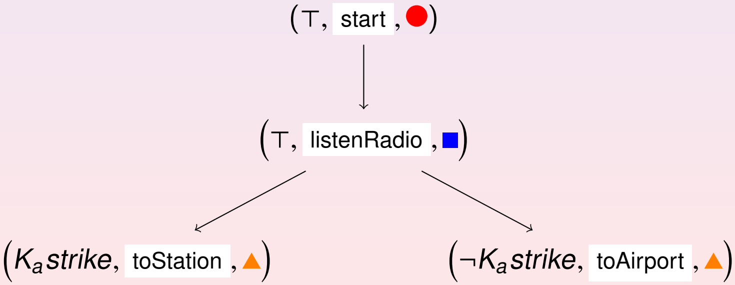
($\top$, listenRadio, ■)

($K_a strike$, toStation, ▲)

($\neg K_a strike$, toAirport, ▲)

Control-flow graph

```
● listenRadio
■if $K_a strike$ then
|        toStation
else
|        toAirport
▲
```



$(\top,$ start $, ●)$

$(\top,$ listenRadio $, ■)$

$(K_a strike,$ toStation $, ▲)$

$(\neg K_a strike,$ toAirport $, ▲)$

Consistent histories (explained with one agent)

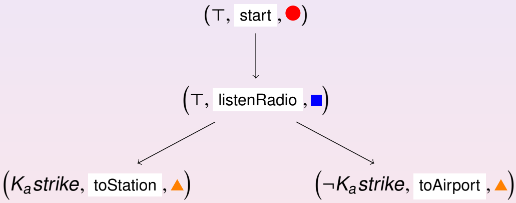
```
● listenRadio
■if $K_a strike$ then
  |      toStation
else
  |      toAirport
▲
```

KBP control-flow graph

In the QdecPOMDP:

$$s^0 \xrightarrow{\text{listenRadio},\ 🔊} s^1$$

$$s^1 \xrightarrow{\text{toStation},\ ☀} s^2$$


$(\top, \text{start}, \bullet)$
$(\top, \text{listenRadio}, \blacksquare)$
$(K_a strike, \text{toStation}, \blacktriangle)$
$(\neg K_a strike, \text{toAirport}, \blacktriangle)$

$$\underbrace{s^0\, (\top, \text{start}, \bullet)\ \boxed{🔊\, s^1 \left(\top, \text{listenRadio}, \blacksquare\right)}}_{\models K_a strike}\ \boxed{☀\, s^2 \left(K_a strike, \text{toStation}, \blacktriangle\right)}$$

Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
   - Verification
   - Execution Problem
   - Succinctness
4. Conclusion

Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
   - Verification
   - Execution Problem
   - Succinctness
4. Conclusion

Verification problem

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.

Verification problem for while-free KBPs

Theorem

The verification problem for while-free KBPs is PSPACE-complete.

Proof idea.

- **Upper bound:** on-the-fly model checking;
- **Lower bound:** reduction from TQBF.

agent 1 value of p_1 agent 2 value of p_2 agent 3 value of p_3

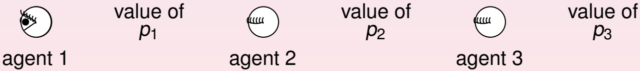
Verification problem for while-free KBPs

Theorem

The verification problem for while-free KBPs is PSPACE-complete.

PROOF IDEA.

- **Upper bound:** on-the-fly model checking;
- **Lower bound:** reduction from TQBF.



agent 1 value of p_1 agent 2 value of p_2 agent 3 value of p_3

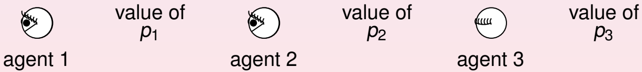
Verification problem for while-free KBPs

Theorem

The verification problem for while-free KBPs is PSPACE-complete.

PROOF IDEA.

- **Upper bound:** on-the-fly model checking;
- **Lower bound:** reduction from TQBF.



Verification problem for while-free KBPs

Theorem

The verification problem for while-free KBPs is PSPACE-complete.

Proof idea.

- **Upper bound:** on-the-fly model checking;
- **Lower bound:** reduction from TQBF.

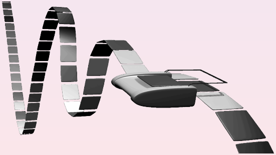
agent 1 value of p_1 agent 2 value of p_2 agent 3 value of p_3

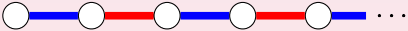
Verification problem for general KBPs

Theorem

The verification problem for general KBPs is undecidable.

PROOF IDEA. Reduction from the halting problem of a Turing machine on input ϵ.





Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
   - Verification
   - Execution Problem
   - Succinctness
4. Conclusion

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)

if justobserved(🔥) **then** turn left **else** stay

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

Proposition

The execution problem for KBPs is PSPACE-complete.

Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
   - Verification
   - Execution Problem
   - Succinctness
4. Conclusion

Modal depth

Modal depth = number of nested ‘... **knows** ’ operators.

Formulas	Modal depths
justobserved(🔥)	0
a **knows** *p*	1
a **knows** (*b* **knows** *p*)	2

Succinctness

Theorem (📄 [Lang, Zanuttini, 2012] for $d = 1$; 📄 [AAAI2018], for $d > 1$)

Let $d \geq 1$.

There is a poly(n)-size QdecPOMDP family $(\mathcal{M}_{n,d})_{n \in \mathbb{N}}$ 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 $poly(n)$-size QdecPOMDP family $(\mathcal{M}_{n,d})_{n \in \mathbb{N}}$ 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.*

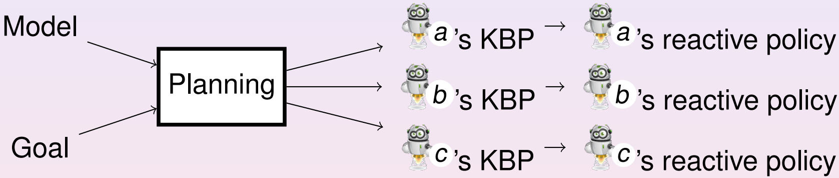
Proof idea. $\mathcal{M}_{n,d}$:

- run a *poly*(n)-time protocol revealing a *poly*(n)-size 3-CNF β;
- β satisfiable iff a d-md non $d - 1$-md expressible epistemic property holds.

Outline

1. Knowledge-based programs
2. Semantics
3. Mathematical Properties
4. Conclusion

Conclusion



Model

Goal

Planning

a's KBP → *a*'s reactive policy

b's KBP → *b*'s reactive policy

c's KBP → *c*'s reactive policy

Higher-order knowledge...

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

Perspectives

- Efficient implementation of the verification/execution problems;
- Heuristics for the planning problem;
- More tractable fragments;

- decPOMDP (with probabilities);
- Temporal properties;
- Strategic reasoning;
- Develop proof systems for KBPs. Use of Coq, Isabelle?