THE EPISTEMIC PLANNING DOMAIN DEFINITION LANGUAGE

Alessandro Burigana
Free University of Bozen-Bolzano, Italy

Francesco Fabiano
University of Parma, Italy

IPS 2022
November 29
Udine, Italy

Epistemic Planning

Epistemic planning is an enrichment of automated (multi-agent) planning where the concept of **knowledge**/**belief** is taken into account:

- Agents might do something depending on **what they know**
- Cooperative setting: agents want to reach a common goal
- Centralized setting: a single omniscient entity (the planner) is responsible for finding a solution

A Simple Running Example

Example (The Letter)

Initial situation. Anne and Bob are in the same room. Anne receives a letter form an university she applied for. The letter states whether she was admitted in the university (u) or not. No one knows whether she was admitted.

There are two possible situations:

- Anne was admitted (u), and
- Anne was not admitted ($\neg u$).

Goals can include **epistemic conditions**:

- Anne knows/believes that u,
- Bob knows/believes that Anne knows/believes whether u or not,
- And so forth.

DYNAMIC EPISTEMIC LOGIC

The Language

Let $\mathcal{P}$ be a finite set of propositional atoms and $\mathcal{AG} = \{1, \ldots, n\}$ a finite set of agents.

Definition (Language $\mathcal{L}^{C}_{\mathcal{P},\mathcal{AG}}$)

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box_i \varphi \mid C_G \varphi,$$

Example (The Letter)

Let $\mathcal{P} = \{c, u\}$ and $\mathcal{AG} = \{Anne, Bob\}$. We can state the conditions of our example as follows:

Initial conditions:

- $\bigwedge_{i \in \mathcal{AG}} (\neg\Box_i u \wedge \neg\Box_i \neg u)$
- $C_{\{Anne, Bob\}} \bigwedge_{i \in \mathcal{AG}} (\neg\Box_i u \wedge \neg\Box_i \neg u)$

Goal conditions:

- $\Box_{Anne} u$
- $\Box_{Bob} (\Box_{Anne} u \vee \Box_{Anne} \neg u)$

The Semantics

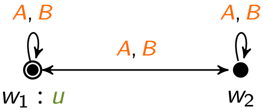


Figure: Initial state.

Epistemic states (pointed Kripke models):

- Worlds: possible situations
- Relations: what agents consider to be possible
- Valuation: what is considered to be true in each world
- Designated worlds: actual situations

The Semantics

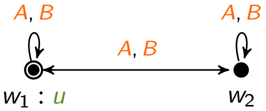


Figure: Initial state.

Epistemic states (pointed Kripke models):

- Worlds: possible situations
- Relations: what agents consider to be possible
- Valuation: what is considered to be true in each world
- Designated worlds: actual situations

Definition (Truth)

$$\begin{array}{lll}
(M, w) \models p & \text{iff} & w \in V(p) \\
(M, w) \models \neg\varphi & \text{iff} & (M, w) \not\models \varphi \\
(M, w) \models \varphi \wedge \psi & \text{iff} & (M, w) \models \varphi \text{ and } (M, w) \models \psi \\
(M, w) \models \Box_i \varphi & \text{iff} & \forall v \text{ if } wR_i v \text{ then } (M, v) \models \varphi \\
(M, w) \models C_G \varphi & \text{iff} & \forall v \text{ if } wR_G^* v \text{ then } (M, v) \models \varphi
\end{array}$$

The Semantics

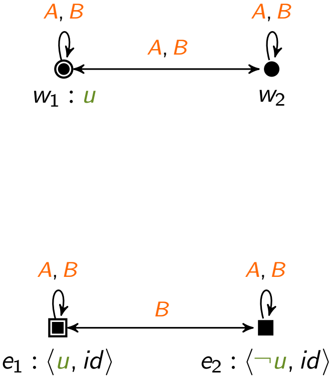
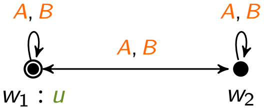


Figure: Anne opens the envelope and reads the letter while Bob is looking. Anne is **fully observant**; Bob is **partially observant**.

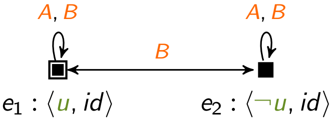
Actions (pointed event models):

- Events: what **might happen** relatively to some agents' perspective
- Relations: akin to those of epistemic models
- Preconditions: what is needed for an event to occur
- Postconditions: how an event changes a world
- Designated events: what actually happens

The Semantics

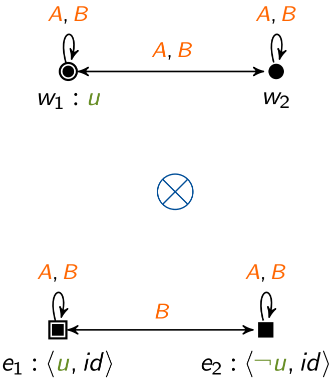


$$\otimes$$



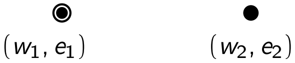
Product update:

The Semantics

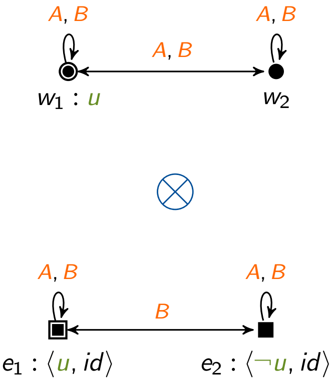


A, B
A, B
A, B
$w_1 : u$
w_2
⊗
A, B
A, B
B
$e_1 : \langle u, id \rangle$
$e_2 : \langle \neg u, id \rangle$

Product update:

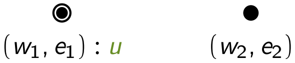


The Semantics

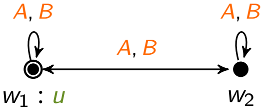


A, B
A, B
A, B
$w_1 : u$
w_2
$\otimes$
A, B
A, B
B
$e_1 : \langle u, id \rangle$
$e_2 : \langle \neg u, id \rangle$

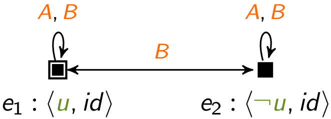
Product update:



The Semantics

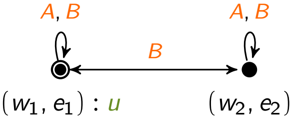


$\otimes$



A, B
A, B
B
$e_1 : \langle u, id \rangle$
$e_2 : \langle \neg u, id \rangle$

Product update:



EPDDL

Why EPDDL?

Main features and motivations:

- Adopts standard PDDL syntax style
  - → Well established language
  - → Shortens the gap between classical planning and epistemic planning representations
  - → Easier to understand even for researchers less familiar with DEL
- Captures the entire DEL semantics
  - → Each component of an event model (events, relations, preconditions, postconditions) is captured by EPDDL
  - → Unified representation of epistemic planning domains: current solvers rely on limited ad hoc languages
  - → Easier comparison between solvers
- Intuitive and usable language
  - → Multiple levels of abstraction (events, action types, actions)
  - → Neat distinction between universal (domain, action type library) and specific (problem) aspects

EPDDL – *Problem*

A **problem** in EPDDL contains the following elements:

- Objects and agents
- Initial state:
  - Explicit representation (worlds, relations, valuation, designated)
  - **Finitary S5 Theory**: desireable theoretical and computational properties
- Goal $\varphi_g \in \mathcal{L}^{C}_{\mathcal{P},\mathcal{AG}}$
  - Propositional formulae are as in PDDL
  - $\Box_i \varphi \rightsquigarrow [\mathtt{i}]\varphi$
  - $C_G \varphi \rightsquigarrow [\mathtt{G}]\varphi$

EPDDL – *Problem* (Example)

```
(define (problem p1)
  (:domain example1)
  (:agents Anne Bob)

  (:init
    (u)
    [Anne Bob](and (not [Anne](u)) (not [Anne](not (u))))
    [Anne Bob](and (not [Bob](u))  (not [Bob](not (u))))
  )

  (:goal
    [Anne](u)
  ))
```

EPDDL – *Problem* (Example)

```
1  (define (problem p1)
2    (:domain example1)
3    (:agents Anne Bob)
4
5    (:init
6      (u)
7      [Anne Bob](and (not [Anne](u)) (not [Anne](not (u))))
8      [Anne Bob](and (not [Bob](u))  (not [Bob](not (u))))
9    )
10
11   (:goal
12     [Anne](u)
13   ))
```

Initial state (lines 5-8):

- u holds
- Anne and Bob have *common knowledge* that Anne *doesn't know whether* u holds
- Anne and Bob have *common knowledge* that Bob *doesn't know whether* u holds

EPDDL – *Problem* (Example)

```
 1 (define (problem p1)
 2   (:domain example1)
 3   (:agents Anne Bob)
 4
 5   (:init
 6     (u)
 7     [Anne Bob](and (not [Anne](u)) (not [Anne](not (u))))
 8     [Anne Bob](and (not [Bob](u))  (not [Bob](not (u))))
 9   )
10
11   (:goal
12     [Anne](u)
13   ))
```

Goal (lines 10-12):

- Anne *knows* that u holds

EPDDL – *Domain* and *Action Type Library*

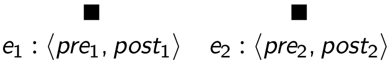
In EPDDL, the universal components are:

- Types (roots of type hierarchy: objects and agents)
- Predicates
- Actions

EPDDL – *Domain* and *Action Type Library*

In EPDDL, the universal components are:

- Types (roots of type hierarchy: objects and agents)
- Predicates
- Actions:
  - **Events**: preconditions and postconditions

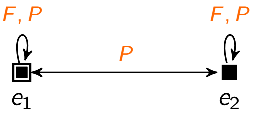


$e_1 : \langle pre_1, post_1 \rangle$ $e_2 : \langle pre_2, post_2 \rangle$

EPDDL – *Domain* and *Action Type Library*

In EPDDL, the universal components are:

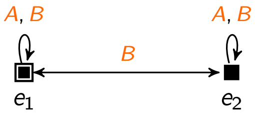
- Types (roots of type hierarchy: objects and agents)
- Predicates
- Actions:
  - Events: preconditions and postconditions
  - **Action type**: observability groups (*e.g.*, *Fully observant*, *Partially observant*), accessibility relations and designated events



EPDDL – *Domain* and *Action Type Library*

In EPDDL, the universal components are:

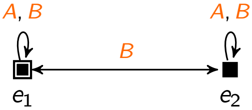
- Types (roots of type hierarchy: objects and agents)
- Predicates
- Actions:
  - Events: preconditions and postconditions
  - Action type: observability groups (*e.g.*, *Fully observant*, *Partially observant*), accessibility relations and designated events
  - **Action**: action type, precondition and observability conditions (Anne is *Fully observant*, Bob is *Partially observant*)



EPDDL – *Domain* and *Action Type Library*

In EPDDL, the universal components are:

- Types (roots of type hierarchy: objects and agents) (**domain**)
- Predicates (**domain**)
- Actions:
  - Events (**action type library**): preconditions and postconditions
  - Action type (**action type library**): observability groups (*e.g.*, *Fully observant*, *Partially observant*), accessibility relations and designated events
  - Action (**domain**): action type, precondition and observability conditions (Anne is *Fully observant*, Bob is *Partially observant*)



Universal components are jointly described by a **domain** and an **action type library**.

EPDDL – *Action Type Library*

Since action types should *not* refer to specific entities of a problem, we implemented two important design choices:

EPDDL – *Action Type Library*

Since action types should *not* refer to specific entities of a problem, we implemented two important design choices:

- **Observability groups**
  - Generalization of accessibility relations
  - Each group represents the perspective of one or more agents

EPDDL – *Action Type Library*

Since action types should *not* refer to specific entities of a problem, we implemented two important design choices:

- Observability groups
  - Generalization of accessibility relations
  - Each group represents the perspective of one or more agents
- **Parametrized events and action types**
  - Abstract from particular predicates and agents
  - Parameters in EPDDL: objects, agents, **formulae** and **postconditions** (*e.g.*, if we pass the preconditions as parameters, we can refer to them as variables within an action type)

EPDDL – *Action Type Library*

Since action types should *not* refer to specific entities of a problem, we implemented two important design choices:

- Observability groups
  - Generalization of accessibility relations
  - Each group represents the perspective of one or more agents
- Parametrized events and action types
  - Abstract from particular predicates and agents
  - Parameters in EPDDL: objects, agents, **formulae** and **postconditions** (*e.g.*, if we pass the preconditions as parameters, we can refer to them as variables within an action type)

Action type libraries can be used transversally across different domains!

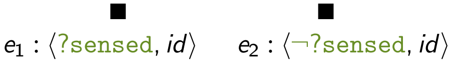
EPDDL – *Action Type Library* (Example)

```
(define (library lib)
  (:event e1
    :parameters (?sensed - predicate)
    :precondition (?sensed))

  (:event e2
    :parameters (?sensed - predicate)
    :precondition (not (?sensed)))

  (:action-type sensing
    :parameters (?p - predicate)
    :observability-groups (Fully Partially)
    :events     (e1 (?sensed :: ?p) )
                (e2 (?sensed :: ?p) )
    :relations  (Fully      (e1 e1) (e2 e2))
                (Partially  (e1 e1) (e2 e2)
                            (e1 e2) (e2 e1))
    :designated (e1)
  )
)
```

Events: e1 and e2



$e_1 : \langle ?\mathtt{sensed}, id \rangle$ $e_2 : \langle \neg ?\mathtt{sensed}, id \rangle$

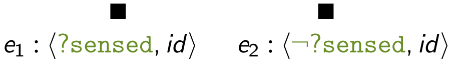
EPDDL – *Action Type Library* (Example)

```
(define (library lib)
  (:event e1
    :parameters (?sensed - predicate)
    :precondition (?sensed))

  (:event e2
    :parameters (?sensed - predicate)
    :precondition (not (?sensed)))

  (:action-type sensing
    :parameters (?p - predicate)
    :observability-groups (Fully Partially)
    :events       (e1 (?sensed :: ?p) )
                  (e2 (?sensed :: ?p) )
    :relations    (Fully      (e1 e1) (e2 e2))
                  (Partially  (e1 e1) (e2 e2)
                              (e1 e2) (e2 e1))
    :designated (e1)
  )
)
```

Events: e1 and e2


$e_1 : \langle$?sensed$, id\rangle$
$e_2 : \langle\neg$?sensed$, id\rangle$

⇓

Action type: sensing

- Parameters: ?sensed :: p

$e_1 : \langle$?p$, id\rangle$
$e_2 : \langle\neg$?p$, id\rangle$

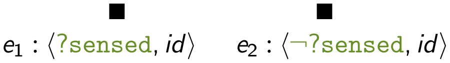
EPDDL – *Action Type Library* (Example)

```
(define (library lib)
  (:event e1
    :parameters (?sensed - predicate)
    :precondition (?sensed))

  (:event e2
    :parameters (?sensed - predicate)
    :precondition (not (?sensed)))

  (:action-type sensing
    :parameters (?p - predicate)
    :observability-groups (Fully Partially)
    :events     (e1 (?sensed :: ?p) )
                (e2 (?sensed :: ?p) )
    :relations  (Fully     (e1 e1) (e2 e2))
                (Partially (e1 e1) (e2 e2)
                           (e1 e2) (e2 e1))
    :designated (e1)
  )
)
```

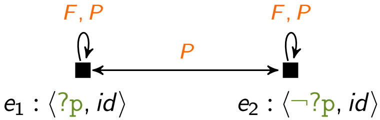
Events: e1 and e2


$e_1 : \langle ?\mathtt{sensed}, id \rangle$
$e_2 : \langle \neg ?\mathtt{sensed}, id \rangle$

$\Downarrow$

Action type: sensing

- Relations: observability groups


F, P
F, P
P
$e_1 : \langle ?\mathtt{p}, id \rangle$
$e_2 : \langle \neg ?\mathtt{p}, id \rangle$

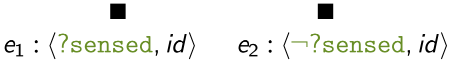
EPDDL – *Action Type Library* (Example)

```
(define (library lib)
  (:event e1
    :parameters (?sensed - predicate)
    :precondition (?sensed))

  (:event e2
    :parameters (?sensed - predicate)
    :precondition (not (?sensed)))

  (:action-type sensing
    :parameters (?p - predicate)
    :observability-groups (Fully Partially)
    :events       (e1 (?sensed :: ?p) )
                  (e2 (?sensed :: ?p) )
    :relations    (Fully      (e1 e1) (e2 e2))
                  (Partially  (e1 e1) (e2 e2)
                              (e1 e2) (e2 e1))
    :designated (e1)
  )
)
```

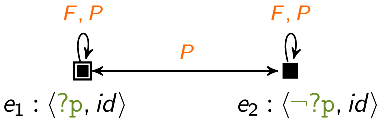
Events: e1 and e2


$e_1 : \langle ?\text{sensed}, id \rangle$
$e_2 : \langle \neg ?\text{sensed}, id \rangle$

$\Downarrow$

Action type: sensing

- Designated: e1 is the designated event


F, P
F, P
P
$e_1 : \langle ?\text{p}, id \rangle$
$e_2 : \langle \neg ?\text{p}, id \rangle$

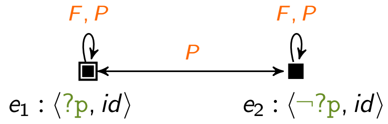
EPDDL – *Domain* (Example)

```
(define (domain example1)
  (:action-type-libraries lib)
  (:requirements :del :typing :equality
                 :universal-conditions)

  (:predicates (u)
               (has_letter ?ag - agent))

  (:action read_letter
    :parameters (?ag - agent)
    :action-type (sensing (?p :: (u)) )
    :precondition (has_letter ?ag)
    :observability-conditions
      (?ag Fully)
      (forall (?ag2 - agent)
          (if (not (= ?ag2 ?ag))
              (Partially)
          ))
  )
)
```

Action type: sensing



F, P
F, P
P
$e_1 : \langle ?p, id \rangle$
$e_2 : \langle \neg ?p, id \rangle$

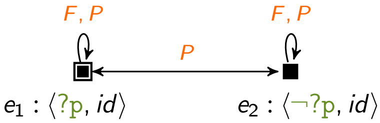
EPDDL – *Domain* (Example)

```
(define (domain example1)
  (:action-type-libraries lib)
  (:requirements :del :typing :equality
                 :universal-conditions)

  (:predicates (u)
               (has_letter ?ag - agent))

  (:action read_letter
    :parameters (?ag - agent)
    :action-type (sensing (?p :: (u)) )
    :precondition (has_letter ?ag)
    :observability-conditions
      (?ag Fully)
      (forall (?ag2 - agent)
          (if (not (= ?ag2 ?ag))
              (Partially)
          ))
  )
)
```

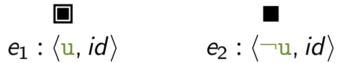
Action type: sensing


F, P
F, P
P
$e_1 : \langle ?p, id \rangle$
$e_2 : \langle \neg ?p, id \rangle$

$\Downarrow$

Action: read_letter A

- Parameters: ?p :: u


$e_1 : \langle u, id \rangle$
$e_2 : \langle \neg u, id \rangle$

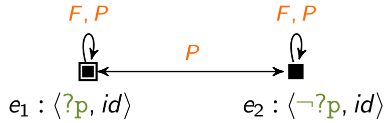
EPDDL – *Domain* (Example)

```
(define (domain example1)
  (:action-type-libraries lib)
  (:requirements :del :typing :equality
                 :universal-conditions)

  (:predicates (u)
               (has_letter ?ag - agent))

  (:action read_letter
    :parameters (?ag - agent)
    :action-type (sensing (?p :: (u)) )
    :precondition (has_letter ?ag)
    :observability-conditions
      (?ag Fully)
      (forall (?ag2 - agent)
          (if (not (= ?ag2 ?ag))
              (Partially)
          ))
  )
)
```

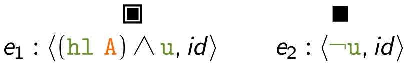
Action type: sensing


F, P
F, P
P
$e_1 : \langle ?\mathtt{p}, id \rangle$
$e_2 : \langle \neg ?\mathtt{p}, id \rangle$

$\Downarrow$

Action: read_letter A

- Precondition: ?has_letter A


$e_1 : \langle (\mathtt{hl\ A}) \wedge \mathtt{u}, id \rangle$
$e_2 : \langle \neg \mathtt{u}, id \rangle$

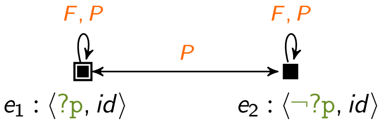
EPDDL – *Domain* (Example)

```
(define (domain example1)
  (:action-type-libraries lib)
  (:requirements :del :typing :equality
                 :universal-conditions)

  (:predicates (u)
               (has_letter ?ag - agent))

  (:action read_letter
    :parameters (?ag - agent)
    :action-type (sensing (?p :: (u)) )
    :precondition (has_letter ?ag)
    :observability-conditions
      (?ag Fully)
      (forall (?ag2 - agent)
          (if (not (= ?ag2 ?ag))
              (Partially)
          ))
  )
)
```

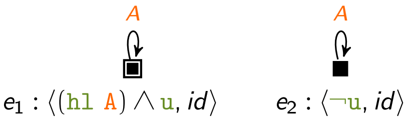
Action type: sensing


F, P
F, P
P
$e_1 : \langle ?\mathrm{p}, id \rangle$
$e_2 : \langle \neg ?\mathrm{p}, id \rangle$

⇓

Action: read_letter A

- Observability: A is Fully


A
A
$e_1 : \langle (\mathrm{hl}\ \mathrm{A}) \land \mathrm{u}, id \rangle$
$e_2 : \langle \neg \mathrm{u}, id \rangle$

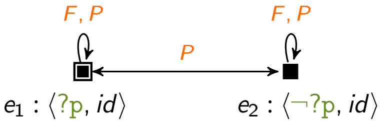
EPDDL – *Domain* (Example)

```
(define (domain example1)
  (:action-type-libraries lib)
  (:requirements :del :typing :equality
                 :universal-conditions)

  (:predicates (u)
               (has_letter ?ag - agent))

  (:action read_letter
    :parameters (?ag - agent)
    :action-type (sensing (?p :: (u)) )
    :precondition (has_letter ?ag)
    :observability-conditions
      (?ag Fully)
      (forall (?ag2 - agent)
          (if (not (= ?ag2 ?ag))
              (Partially)
          ))
  )
)
```

Action type: sensing

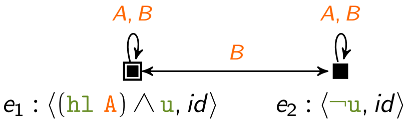


F, P
F, P
P
$e_1 : \langle ?\mathrm{p}, id \rangle$
$e_2 : \langle \neg ?\mathrm{p}, id \rangle$

$\Downarrow$

Action: read_letter A

- Observability: B is Partially



A, B
A, B
B
$e_1 : \langle (\mathrm{hl}\ \mathrm{A}) \wedge \mathrm{u}, id \rangle$
$e_2 : \langle \neg \mathrm{u}, id \rangle$

FUTURE WORKS

Future Works

- Finalizing the last details (we are open to your suggestions!)
- Implementing full-fledged parser (with type checker)
- Creating a public repository of epistemic planning domains to be used as benchmarks for epistemic planners
- Creating a public Wiki page for EPDDL

THANK YOU

Questions?