EPISTEMIC PLANNING:
RECENT ADVANCEMENTS AND FUTURE DIRECTIONS

Alessandro Burigana
Free University of Bozen-Bolzano

November 11th, 2025

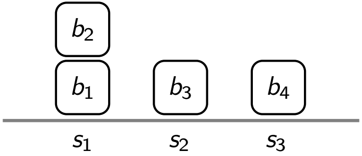
The 35th International Conference
on Automated Planning and Scheduling,

Melbourne, Victoria, Australia

Example (Blocks World)

- An initial configuration of blocks are piled up in stacks is given;
- The agent can move one block (at a time) from the top of a stack to another;
- From an initial configuration, the agent must move the blocks to achieve a desired one.

Initial state:



Actions $\mathit{move}(b, x, y)$:

- $\text{Pre}(\mathit{move}(b, x, y)) = On(b, x) \wedge Clear(b) \wedge Clear(y)$
- $\text{Eff}(\mathit{move}(b, x, y)) = \{On(b, y), Clear(x), \neg On(b, x), \neg Clear(y)\} \triangleright \top$

Epistemic Planning

Epistemic planning: enrichment of classical planning with notions of knowledge and belief.

- → **Epistemic states** represent what the agents know/believe about the world and others' perspective of the world.
- → **Epistemic actions** can change both the world and the knowledge/belief of the agents.
- → Agents have to reason about each others' (higher-order) knowledge/beliefs to reach a shared goal.
- → We move from a propositional, single-agent, fully observable, deterministic setting to an modal, multi-agent, partially observable, non-deterministic one.

Semantics for Epistemic Planning

We can define two main families of semantics for epistemic planning:

1. **Sentential approaches**:
   - **Epistemic states**: sets of formulas called knowledge (or belief) bases.
   - **Epistemic actions**: typically allow to modify a state by adding/deleting epistemic formulas (akin to classical actions).
2. **Dynamic Epistemic Logic**:
   - **Epistemic states**: pointed Kripke models, where a set of possible worlds represents different perspectives of agents about a situation.
   - **Epistemic actions**: pointed event models, where a set of possible events represents different agents' view of some information change.

EPISTEMIC LOGIC

Syntax

Let P be a finite set of propositional atoms and $Ag = \{1, \ldots, n\}$ a finite set of agents. The **language** $\mathcal{L}_{P,Ag}$ **of Epistemic Logic** is given by the BNF:

Definition (Language of Epistemic Logic)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \Box_i \phi,$$

→ Operator $\Box_i$: depending on the context, describes what agent i knows or believes.

→ Dual operator $\Diamond_i$ ($\equiv \neg\Box_i\neg$): describes what agent i considers to be possible or compatible.

An **epistemic state** represents both factual information and what agents know/believe.

Definition (Epistemic Model)

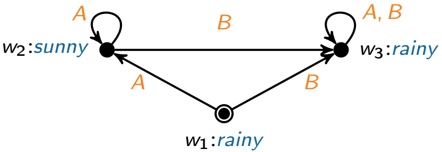
An **epistemic model** is a triple $M = (W, R, L)$, where:

- $W \neq \varnothing$ is a finite set of **possible worlds**;
- $R : Ag \to 2^{W \times W}$ assigns to each agent i an **accessibility relation** R_i; and
- $L : W \to 2^P$ assigns to each world a **label**, being a finite set of atoms.

Definition (Epistemic State)

An **epistemic state** is a pair (M, w_d), where $w_d \in W$ is the **designated world**.

Example

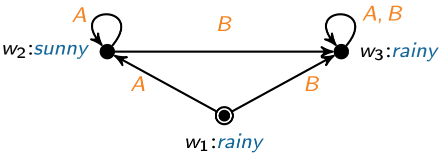


Definition (Truth)

Let $s = (M, w_d)$, where $M = (W, R, L)$, be an **epistemic state** and let $w \in W$:

$$
\begin{aligned}
(M, w) &\models p && \text{iff} \quad p \in L(w) \\
(M, w) &\models \neg\phi && \text{iff} \quad (M, w) \not\models \phi \\
(M, w) &\models \phi \wedge \psi && \text{iff} \quad (M, w) \models \phi \text{ and } (M, w) \models \psi \\
(M, w) &\models \Box_i \phi && \text{iff} \quad \forall v \text{ if } wR_i v \text{ then } (M, v) \models \phi
\end{aligned}
$$

Example



- $\Box_{Anne}\, sunny$
- $\Box_{Bob}\, rainy$
- $\Box_{Anne}\Box_{Bob}\, rainy$
- $\Diamond_{Bob}\Box_{Anne}\, rainy$

To Know or to Believe?

How can epistemic states represent the knowledge and the beliefs of agents?

$\rightarrow$ We model them via **axioms**.

	Axiom	Frame Property	Knowledge	Belief
K	$\Box_i(\phi \rightarrow \psi) \rightarrow (\Box_i\phi \rightarrow \Box_i\psi)$	-	✓	✓
T	$\Box_i\phi \rightarrow \phi$	Reflexivity	✓	
D	$\Box_i\phi \rightarrow \Diamond_i\phi$	Seriality	✓	✓
4	$\Box_i\phi \rightarrow \Box_i\Box_i\phi$	Transitivity	✓	✓
5	$\neg\Box_i\phi \rightarrow \Box_i\neg\Box_i\phi$	Euclideanness	✓	✓

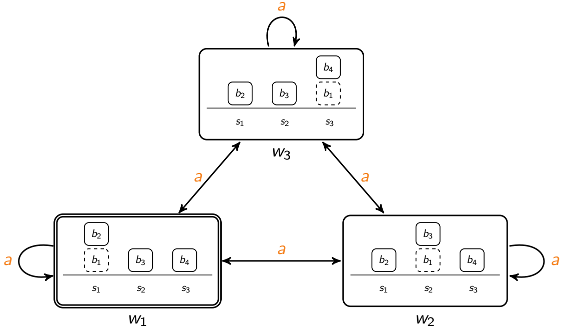
An epistemic state represents:

- Knowledge, when it satisfies axioms **K**, **T**, **4** and **5** $\Rightarrow$ **Logic S5**$_n$
- Belief, when it satisfies axioms **K**, **D**, **4** and **5** $\Rightarrow$ **Logic KD45**$_n$

Epistemic Blocks World

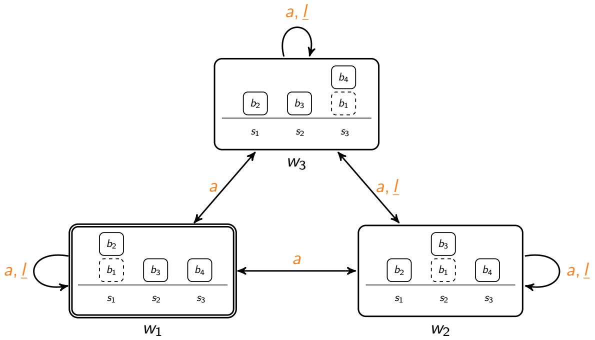
Example (Epistemic Blocks World)

- Agent a: only sees from above.



Example (Multi-Agent Epistemic Blocks World)

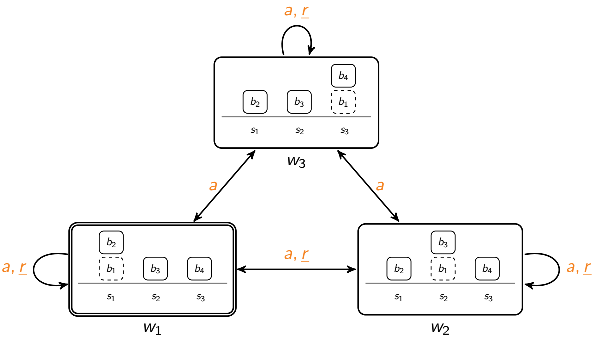
- Agent a: only sees from above.
- Agent l: only sees from a top left position.



Epistemic Blocks World

Example (Multi-Agent Epistemic Blocks World)

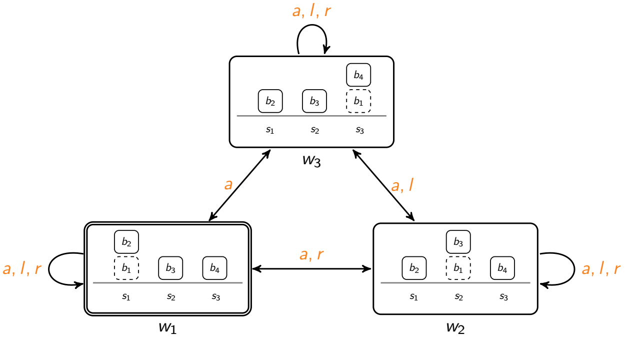
- Agent *a*: only sees from above.
- Agent *r*: only sees from a top right position.



Epistemic Blocks World

Example (Multi-Agent Epistemic Blocks World)

- Agent a: only sees from above.
- Agent l: only sees from a top left position.
- Agent r: only sees from a top right position.



DYNAMIC EPISTEMIC LOGIC

Epistemic Actions

Definition (Event Model)

An **event model** is a quadruple $A = (E, Q, pre, post)$, where:

- $E \neq \varnothing$ is a finite set of **events**;
- $Q : Ag \to 2^{E \times E}$ assigns to each agent i an **accessibility relation** Q_i;

Intuitively:

- An event can be seen as a classical action.
- Accessibility relations specify the perspectives of agents on which events take place.

Definition (Event Model)

An **event model** is a quadruple $A = (E, Q, pre, post)$, where:

- $E \neq \varnothing$ is a finite set of **events**;
- $Q : Ag \to 2^{E \times E}$ assigns to each agent i an **accessibility relation** Q_i;
- $pre : E \to \mathcal{L}_{P,Ag}$ assigns to each event a **precondition**;
- $post : E \times P \to \mathcal{L}_{P,Ag}$ assigns to each event-atom pair a **postcondition**.

Intuitively:

- An event can be seen as a classical action, **each with its own pre- and postconditions**.
- Accessibility relations specify the perspectives of agents on which events take place.

Epistemic Actions

Definition (Event Model)

An **event model** is a quadruple $A = (E, Q, pre, post)$, where:

- $E \neq \varnothing$ is a finite set of **events**;
- $Q : Ag \to 2^{E \times E}$ assigns to each agent i an **accessibility relation** Q_i;
- $pre : E \to \mathcal{L}_{P,Ag}$ assigns to each event a **precondition**;
- $post : E \times P \to \mathcal{L}_{P,Ag}$ assigns to each event-atom pair a **postcondition**.

Intuitively:

- An event can be seen as a classical action, **each with its own pre- and postconditions**.
- Accessibility relations specify the perspectives of agents on which events take place.

Definition (Epistemic Action)

An **epistemic action** is a pair (A, e_d), where $e_d \in E$ is the **designated event**.

Product Update

An action (A, e_d) is **applicable** is an epistemic state (M, w_d) iff $(M, w_d) \models pre(e_d)$.

Definition (Product Update)

Given (M, w_d) and (A, e_d), where $M = (W, R, L)$ and $A = (E, Q, pre, post)$, their **product update** $(M, w_d) \otimes (A, e_d)$ is the epistemic state $((W', R', L'), w'_d)$ where:

Product Update

An action (A, e_d) is **applicable** is an epistemic state (M, w_d) iff $(M, w_d) \models pre(e_d)$.

Definition (Product Update)

Given (M, w_d) and (A, e_d), where $M = (W, R, L)$ and $A = (E, Q, pre, post)$, their **product update** $(M, w_d) \otimes (A, e_d)$ is the epistemic state $((W', R', L'), w'_d)$ where:

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;

An action (A, e_d) is **applicable** is an epistemic state (M, w_d) iff $(M, w_d) \models pre(e_d)$.

Definition (Product Update)

Given (M, w_d) and (A, e_d), where $M = (W, R, L)$ and $A = (E, Q, pre, post)$, their **product update** $(M, w_d) \otimes (A, e_d)$ is the epistemic state $((W', R', L'), w'_d)$ where:

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;

Product Update

An action (A, e_d) is **applicable** is an epistemic state (M, w_d) iff $(M, w_d) \models pre(e_d)$.

Definition (Product Update)

Given (M, w_d) and (A, e_d), where $M = (W, R, L)$ and $A = (E, Q, pre, post)$, their **product update** $(M, w_d) \otimes (A, e_d)$ is the epistemic state $((W', R', L'), w'_d)$ where:

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and

Product Update

An action (A, e_d) is **applicable** is an epistemic state (M, w_d) iff $(M, w_d) \models pre(e_d)$.

Definition (Product Update)

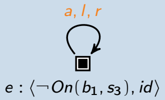
Given (M, w_d) and (A, e_d), where $M = (W, R, L)$ and $A = (E, Q, pre, post)$, their **product update** $(M, w_d) \otimes (A, e_d)$ is the epistemic state $((W', R', L'), w'_d)$ where:

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

Example

Public Announcement

Agent r publicly tells everybody that $\neg On(b_1, s_3)$.



Example

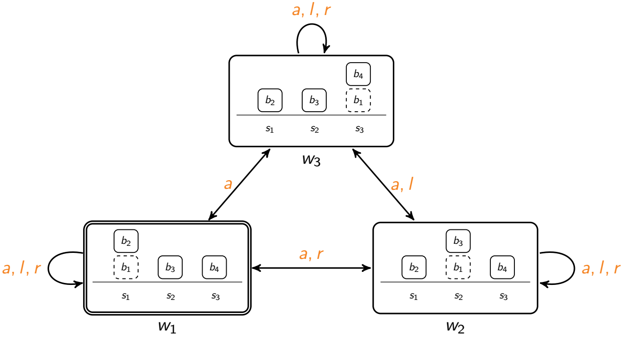
Public Announcement

Agent r publicly tells everybody that $\neg On(b_1, s_3)$.

$e : \langle \neg On(b_1, s_3), id \rangle$

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



Example

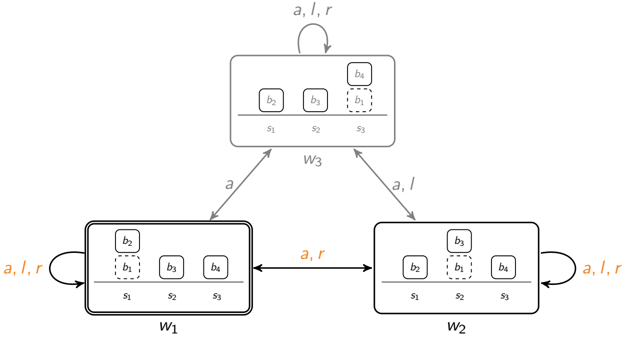
Public Announcement

Agent r publicly tells everybody that $\neg On(b_1, s_3)$.

$e : \langle \neg On(b_1, s_3), id \rangle$

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



Example

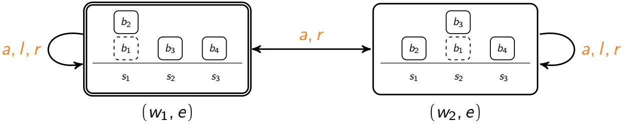
Public Announcement

Agent r publicly tells everybody that $\neg On(b_1, s_3)$.

$e : \langle \neg On(b_1, s_3), id \rangle$

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



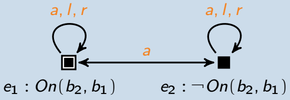
Semi-Private Sensing Action

Example

Semi-Private Sensing Action

Agent r peeks under block b_2 while agents a and l observe him. Specifically:

- Agents r and l observe what is actually being sensed.
- Agent a can not directly observe what agent r is seeing.



Trivial postconditions are omitted.

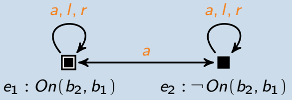
Semi-Private Sensing Action

Example

Semi-Private Sensing Action

Agent r peeks under block b_2 while agents a and l observe him. Specifically:

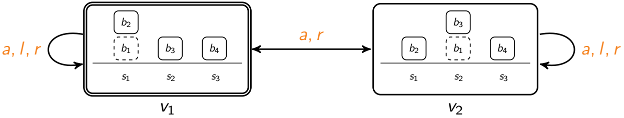
- Agents r and l observe what is actually being sensed.
- Agent a can not directly observe what agent r is seeing.



Trivial postconditions are omitted.

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



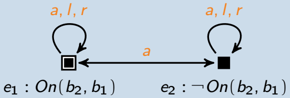
Semi-Private Sensing Action

Example

Semi-Private Sensing Action

Agent r peeks under block b_2 while agents a and l observe him. Specifically:

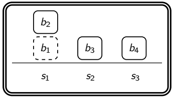
- Agents r and l observe what is actually being sensed.
- Agent a can not directly observe what agent r is seeing.



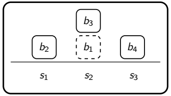
Trivial postconditions are omitted.

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



(v_1, e_1)



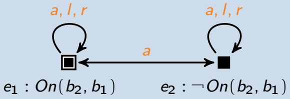
(v_2, e_2)

Example

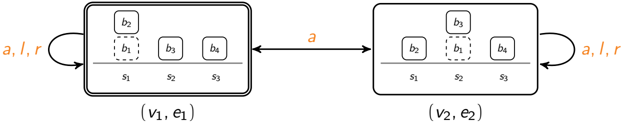
Semi-Private Sensing Action

Agent r peeks under block b_2 while agents a and l observe him. Specifically:

- Agents r and l observe what is actually being sensed.
- Agent a can not directly observe what agent r is seeing.



Trivial postconditions are omitted.



Definition (Product Update)

- $W' = \{(w,e) \in W \times E \mid (M,w) \models pre(e)\}$;
- $R'_i = \{((w,e),(v,f)) \in W' \times W' \mid wR_iv \text{ and } eQ_if\}$;
- $L'((w,e)) = \{p \in P \mid (M,w) \models post(e,p)\}$; and
- $w'_d = (w_d, e_d)$.

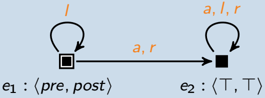
Private Ontic Actions

Example

Private Ontic Action

Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



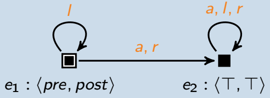
Private Ontic Actions

Example

Private Ontic Action

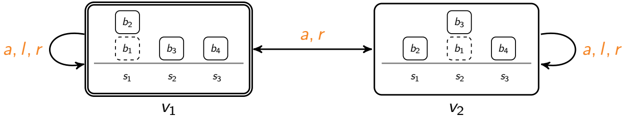
Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

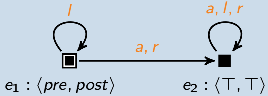


Example

Private Ontic Action

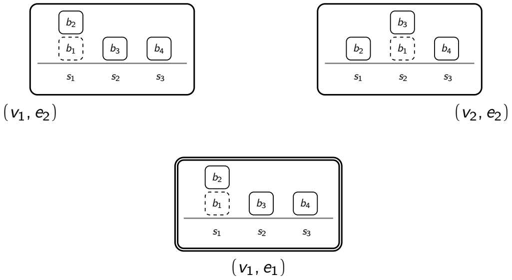
Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

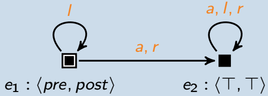


Example

Private Ontic Action

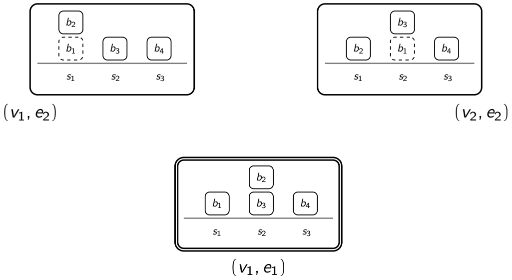
Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

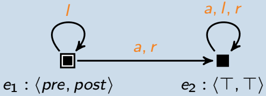


Example

Private Ontic Action

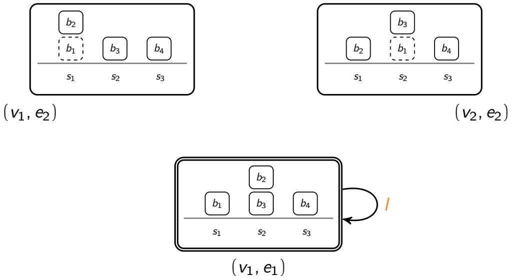
Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



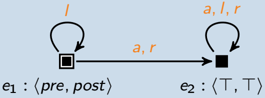
Private Ontic Actions

Example

Private Ontic Action

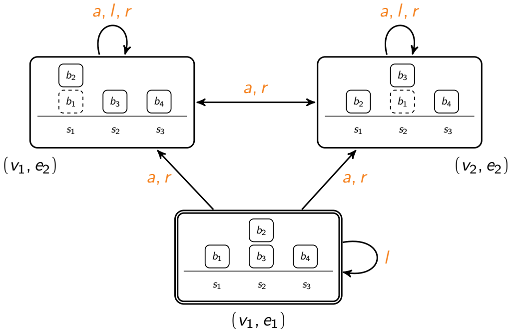
Agent l privately moves block b_2 from b_1 to b_3, where:

- $pre = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_3)$
- $post(e_1, On(b_2, b_1)) = \bot$
- $post(e_1, On(b_2, b_3)) = \top$



Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.



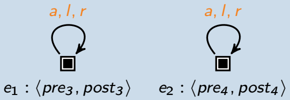
A Glance at Non-Deterministic Actions

Example

Public Non-Deterministic Ontic Action

Agents publicly flip a coin: if heads, they move b_2 from b_1 to b_3, otherwise to b_4, where:

- $pre_k = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_k)$
- $post_k(e_1, On(b_2, b_1)) = \bot$
- $post_k(e_1, On(b_2, b_k)) = \top$



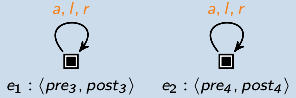
A Glance at Non-Deterministic Actions

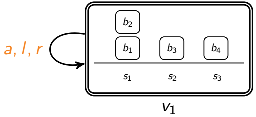
Example

Public Non-Deterministic Ontic Action

Agents publicly flip a coin: if heads, they move b_2 from b_1 to b_3, otherwise to b_4, where:

- $pre_k = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_k)$
- $post_k(e_1, On(b_2, b_1)) = \bot$
- $post_k(e_1, On(b_2, b_k)) = \top$





Definition (Product Update)

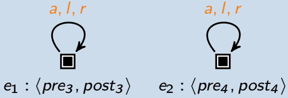
- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

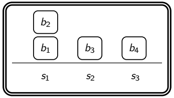
Example

Public Non-Deterministic Ontic Action

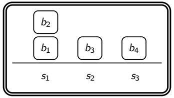
Agents publicly flip a coin: if heads, they move b_2 from b_1 to b_3, otherwise to b_4, where:

- $pre_k = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_k)$
- $post_k(e_1, On(b_2, b_1)) = \bot$
- $post_k(e_1, On(b_2, b_k)) = \top$





(v_1, e_1)



(v_1, e_2)

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

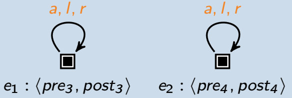
A Glance at Non-Deterministic Actions

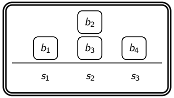
Example

Public Non-Deterministic Ontic Action

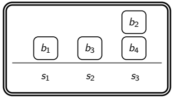
Agents publicly flip a coin: if heads, they move b_2 from b_1 to b_3, otherwise to b_4, where:

- $pre_k = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_k)$
- $post_k(e_1, On(b_2, b_1)) = \bot$
- $post_k(e_1, On(b_2, b_k)) = \top$





(v_1, e_1)



(v_1, e_2)

Definition (Product Update)

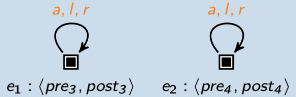
- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v \text{ and } eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

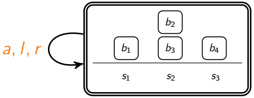
Example

Public Non-Deterministic Ontic Action

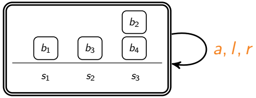
Agents publicly flip a coin: if heads, they move b_2 from b_1 to b_3, otherwise to b_4, where:

- $pre_k = On(b_2, b_1) \wedge Clear(b_2) \wedge Clear(b_k)$
- $post_k(e_1, On(b_2, b_1)) = \bot$
- $post_k(e_1, On(b_2, b_k)) = \top$





(v_1, e_1)



(v_1, e_2)

Definition (Product Update)

- $W' = \{(w, e) \in W \times E \mid (M, w) \models pre(e)\}$;
- $R'_i = \{((w, e), (v, f)) \in W' \times W' \mid wR_i v$ and $eQ_i f\}$;
- $L'((w, e)) = \{p \in P \mid (M, w) \models post(e, p)\}$; and
- $w'_d = (w_d, e_d)$.

The Epistemic Plan Existence Problem

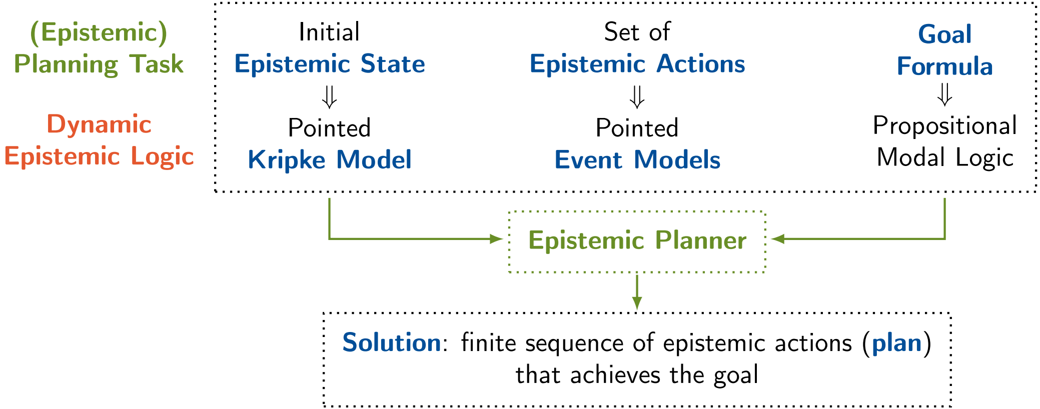
(Epistemic) Planning Task

Initial **Epistemic State**	Set of **Epistemic Actions**	**Goal Formula**

The Epistemic Plan Existence Problem

	Initial	Set of	
(Epistemic) Planning Task	**Epistemic State**	**Epistemic Actions**	**Goal Formula**
	⇓	⇓	⇓
Dynamic Epistemic Logic	Pointed **Kripke Model**	Pointed **Event Models**	Propositional Modal Logic

The Epistemic Plan Existence Problem



The Epistemic Plan Existence Problem

(Epistemic) Planning Task

Dynamic Epistemic Logic

	Initial **Epistemic State**	Set of **Epistemic Actions**	**Goal Formula**
	⇓	⇓	⇓
	Pointed **Kripke Model**	Pointed **Event Models**	Propositional Modal Logic

→ **Epistemic Planner** ←

↓

Solution: finite sequence of epistemic actions (**plan**) that achieves the goal

Epistemic Plan Existence Problem

Given an epistemic planning task, **does there exist a plan that achieves the goal**?

Classical Vs. Epistemic Actions

Classical planning:

1. **Propositional**
2. **Single-agent**
3. **Fully Observable**
4. **Deterministic**
5. **Ontic change**

Epistemic planning:

1. **Modal**
2. **Multi-agent**
3. **Partially Observable**
4. **Non-deterministic** (multi-pointed models are needed)
5. **Ontic and epistemic change**

Moreover, agents can reason on higher-order knowledge/beliefs of others to any nesting level.

$\rightarrow$ There are **no bounds** on the reasoning power of agents!

Theorem (Bolander and Andersen [BA11])

The epistemic plan existence problem is ***undecidable****.*

RECENT ADVANCEMENTS

Several Different Directions

Many approaches have been pursued in epistemic planning. Today we cover the following:

1. Sentential approaches:
   - Compilations to classical planning.
   - **Alternating cover disjunctive formulas**.
2. Heuristics for epistemic planning.
3. DEL-based approaches:
   - (Bounded) **bisimulation contractions**.
   - **Depth-bounded** epistemic planning.

SENTENTIAL APPROACHES

Compilations to Classical Planning

PDDL translation by Kominis and Geffner [KG15] of the next epistemic planning formalism:

- **Public ontic actions**: all agents know both about the action and its effects.
- **Semi-private sensing/announcements actions**: all agents know about the action, but only some know its effects.

They show that:

- The compilation is quadratic.
- The identified fragment is **PSPACE-complete**.
- Their formalism corresponds to a DEL fragment.

Compilations to Classical Planning (cont.)

PDDL translation by Muise et al. [Mui+15; Mui+22] based on a restricted language:

Definition (Restricted Modal Literals)

$$\phi ::= p \mid \neg\phi \mid \Box_i \phi$$

- **Epistemic states**: **sets of RMLs**.
- **Epistemic actions**: **preconditions/effects** pairs defined over RMLs.
- **Promising results in different epistemic planning benchmarks**.
- Worse performances on instances with higher reasoning depth.
- More expressive than Kominis and Geffner's approach (*e.g.*, allows for private actions).

Compilations to Classical Planning (cont.)

A similar approach is pursued by Cooper et al. [Coo+16], later generalized by [Coo+20]:

Definition (Epistemic Logic of Observation (EL-O))

$$\alpha ::= p \mid S_i \alpha \mid JS\alpha$$
$$\phi ::= \alpha \mid \neg\phi \mid \phi \wedge \phi$$

where $S_i\phi$ means that agent i **sees whether** ϕ holds and $JS\phi$ means that all agents **jointly see** whether ϕ.

→ $\phi \wedge S_i\phi$ is equivalent to $\Box_i\phi$.

→ $\phi \wedge JS\phi$ is equivalent to $C\phi$ (common knowledge of ϕ).

- They show that the problem is **PSPACE-complete**.
- More expressive than Muise et al.'s approach (allows for common knowledge and parallel actions).

Pros and Cons

Pros

- Rely on **efficiency** of classical planners.
- **Lower complexity** of the plan existence problem.

Cons

- **Limited** to specific fragments.
- Typically **do not scale well** when higher-order knowledge is involved.

Alternating Cover Disjunctive Formulas

Huang et al. [Hua+17] proposed a doxastic planning framework, *i.e.*, based on the logic of belief $KD45_n$:

- **Epistemic states** are general $KD45_n$ formulas with common knowledge.
- **Deterministic actions**: precondition/effects pairs over $KD45_n$ formulas.
- **Sensing actions**: precondition + positive and negative effects over $KD45_n$ formulas.
- Formulas are transformed into equivalent **Alternating Cover Disjunctive Formulas** (ACDF).
  - → Length of an ACDF formula is shown to be **at most singly exponential** in the length of the original formula.

Alternating Cover Disjunctive Formulas: Pros and Cons

- A **Pruning AND-OR** (PrAO) search algorithm with visited state check is provided.
- The algorithm builds an action tree branching on sensing actions.
- Here a stronger notion of equivalence of ACDF formulas is introduced, which can be checked in polynomial time.
- The planner, called **MEPK**, is compared to the solvers by Kominis and Geffner, and by Muise et al.

Pros

- The formalism is **more expressive** than the compilation-based ones.
- **Reasonable** performances on the conducted experiments.

Cons

- **Worse performances** compared to compilation-based approaches.
- **Exponential blowup** of ACDF formulas size.

HEURISTICS FOR EPISTEMIC PLANNING

Heuristics for MEPK

Later, the MEPK planner was improved as follows [Wu18]:

- A normal form for ACDF formulas is provided, called **ADNF**, which is claimed to be more space efficient than regular ACDF.
- A notion of **distance** is provided for ADNF formulas, used to guide the search towards states with lower distance from the goal.
- Two heuristic strategies for pruning the search space are also provided.

→ The resulting planner, called **MEPL**, was benchmarked against MEPK, showing improvements on the vast majority of the tested instances.

More Heuristics for MEPK

Heuristics for MEPK have been also developed in a subsequent work [FL24]:

- **Enhancement**: use information in the path leading to the first goal-satisfying state to guide the rest of the search.
- **Belief lock**: in some cases, once an agent has acquired some belief, it can not later forget it (the belief is "locked").
  - → A set of conditions is identified for **recognizing locked beliefs**.
  - → Belief locks are used for **pruning** the search space.
  - → For instance, if in the current state $\Box_i p$ is recognized as a locked belief, and the goal requires that $\Box_i \neg p$ instead, we can safely prune the search, as the goal is unreachable.

→ The comparison with the original MEPK planner showed performance improvements. However, no comparison with MEPL was conducted.

Heuristics for EFP 2.0

EFP 2.0 was later equipped with heuristic search strategies [Fab+24]

- **Several heuristics** were proposed, based on planning graph methods and on maximal goal sub-formulas satisfaction.
- A **portfolio-like technique** was used to construct a machine learning model for selecting the best heuristic for each input problem.

Preliminary experiments showed the following:

- 👍 General **improvements** over EFP 2.0 with no heuristics.
- 👎 Minimality of plans **not guaranteed**.

DEL-BASED APPROACHES

Main Challenges

- **Higher uncertainty** of agents means **bigger models**.
- Worst-case **exponential blowup** of size of states after product update.
- **Expensive check** for visited states.
- Search space can be infinite.

Bisimulations

- A **bisimulation** between two states s and s' is a binary relation Z on their world-sets s.t.:
  - → If $(x, x') \in Z$, then x and x' are **propositionally equivalent** (**atom**) and **for each** i**-successor** y of x **there exists an** i**-successor** y' of x' **s.t.** $(x', y') \in Z$ (**forth**), and vice versa (**back**).
- If such a Z exists, we say that **s and s'** are **bisimilar**, written $s \leftrightarrows s'$.
- Bisimilarity corresponds to **modal equivalence**:

Proposition ([BRV01])

Two states are ***bisimilar*** *iff they* ***satisfy the same formulas in*** $\mathcal{L}_{P,Ag}$.

- A **bisimulation** between two states s and s' is a binary relation Z on their world-sets s.t.:
  - → If $(x, x') \in Z$, then x and x' are **propositionally equivalent** (**atom**) and **for each** i**-successor** y of x **there exists an** i**-successor** y' of x' **s.t.** $(x', y') \in Z$ (**forth**), and vice versa (**back**).
- If such a Z exists, we say that s **and** s' are **bisimilar**, written $s \leftrightarroweq s'$.
- Bisimilarity corresponds to **modal equivalence**:

Proposition ([BRV01])

*Two states are **bisimilar** iff they **satisfy the same formulas in** $\mathcal{L}_{P,Ag}$.*

Example (Two bisimilar states)

Bisimulations

- A **bisimulation** between two states s and s' is a binary relation Z on their world-sets s.t.:
  - → If $(x, x') \in Z$, then x and x' are **propositionally equivalent** (**atom**) and **for each** i**-successor** y of x **there exists an** i**-successor** y' of x' **s.t.** $(x', y') \in Z$ (**forth**), and vice versa (**back**).
- If such a Z exists, we say that **s and s'** are **bisimilar**, written $s \leftrightarroweq s'$.
- Bisimilarity corresponds to **modal equivalence**:

Proposition ([BRV01])

Two states are ***bisimilar*** *iff they* ***satisfy the same formulas in*** $\mathcal{L}_{P,Ag}$.

Proposition (Product Update Preserves Bisimilarity [DHK07])

If $s \leftrightarroweq s'$ and α is applicable in both, then:

$$s \otimes \alpha \leftrightarroweq s' \otimes \alpha$$

Reducing the Size of Visited States

Definition (Bisimulation Contraction)

The **(bisimulation) contraction** of s is the **quotient structure** $\lfloor s \rfloor_{\leftrightarrow}$ of s induced by the bisimilarity relation.

Proposition ([BRV01])

*$\lfloor s \rfloor_{\leftrightarrow}$ is a **minimal state** (smallest number of worlds and edges) **bisimilar** to s.*

Reducing the Size of Visited States

Definition (Bisimulation Contraction)

The **(bisimulation) contraction** of s is the **quotient structure** $\lfloor s \rfloor_{\leftrightarrow}$ of s induced by the bisimilarity relation.

Proposition ([BRV01])

*$\lfloor s \rfloor_{\leftrightarrow}$ is a **minimal state** (smallest number of worlds and edges) **bisimilar** to s.*

Key Idea

We can **replace any visited state s with** its bisimulation contraction $\lfloor s \rfloor_{\leftrightarrow}$.

- $\lfloor s \rfloor_{\leftrightarrow}$ and s are bisimilar, and so are $\lfloor s \rfloor_{\leftrightarrow} \otimes \alpha$ and $s \otimes \alpha$.
- The **size of $\lfloor s \rfloor_{\leftrightarrow}$ is at most the size of s**.
- We compute and store less information.
- Technique adopted by several epistemic planners [Fab+20; BDH21; BBM25].

EFP and PG-EFP

One of the earlier DEL-based planners is the **Epistemic Forward Planner** [Le+18], based on the $m\mathcal{A}^*$ epistemic action description language [Bar+15; Bar+22]:

- **Private/public ontic actions.**
- **Public/(Semi-)-private sensing/announcements actions.**

Two search strategies were implemented:

- **EFP**: Breadth-First Search.
- **PG-EFP**: **planning graph heuristic** tailored for the $m\mathcal{A}^*$ fragment.

EFP 2.0

Later, Fabiano et al. [Fab+20] implemented an improved version of the planner, called **EFP 2.0**.

Two new search algorithms

- Kripke-based BFS search with **bisimulation contractions** and check for visited states.
- BFS search based on an alternative semantics for epistemic states called **possibilities**.
  - More **compact representation** of states.
  - Natural **reuse of previously calculated information**.

EFP 2.0

Later, Fabiano et al. [Fab+20] implemented an improved version of the planner, called **EFP 2.0**.

Two new search algorithms

- Kripke-based BFS search with **bisimulation contractions** and check for visited states.
- BFS search based on an alternative semantics for epistemic states called **possibilities**.
  - More **compact representation** of states.
  - Natural **reuse of previously calculated information**.

Experiments results

- Improved performances wrt. EFP 1.0, especially the possibility-based planner.
- Promising results in many epistemic planning benchmarks.
- Muise et al. [Mui+22] compared to EFP 2.0
  - They showed **better performances** than EFP 2.0 on **smaller instances**.
  - And **worse** results on instances that required **higher reasoning depth**.

Pros and Cons

Pros

- **Efficient** running times for the considered fragment.
- **Good scalability** on bigger instances.

Cons

- **Limited** to specific fragments.
- **Check for visited states** is computationally **expensive**.

Improving Check for Visited States

How do we check if a state s has already been visited? Suppose we have a set *Visited* of visited states:

Improving Check for Visited States

How do we check if a state s has already been visited? Suppose we have a set *Visited* of visited states:

1. Simply check if $s \in \mathit{Visited}$.

Improving Check for Visited States

How do we check if a state s has already been visited? Suppose we have a set *Visited* of visited states:

1. ~~Simply check if $s \in \mathit{Visited}$.~~
2. For all $t \in \mathit{Visited}$, check if $s \leftrightarrows t$.

Improving Check for Visited States

How do we check if a state s has already been visited? Suppose we have a set *Visited* of visited states:

1. ~~Simply check if $s \in \mathit{Visited}$.~~
2. For all $t \in \mathit{Visited}$, check if $s \leftrightarrows t$.

   $\rightarrow$ Works, but **very expensive**.

Improving Check for Visited States

How do we check if a state s has already been visited? Suppose we have a set *Visited* of visited states:

1. ~~Simply check if $s \in Visited$.~~
2. For all $t \in Visited$, check if $s \leftrightarrows t$.
   → Works, but **very expensive**.

Can we do better than this?

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

- Start from an initial partition of W calculated wrt. labels.

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

- Start from an initial partition of W calculated wrt. labels.
- At each step, iterate through the elements, called **blocks**, B of the partition.

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

- Start from an initial partition of W calculated wrt. labels.
- At each step, iterate through the elements, called **blocks**, B of the partition.
- For $x \in B$ and $x' \in B'$, we say that B **sees** B' **via agent** i **iff** xR_ix'.

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

- Start from an initial partition of W calculated wrt. labels.
- At each step, iterate through the elements, called **blocks**, B of the partition.
- For $x \in B$ and $x' \in B'$, we say that B **sees** B' **via agent** i **iff** xR_ix'.
- For all agents $i \in Ag$, split B into sub-blocks $B_1, \ldots, B_k$ such that the worlds in each sub-block see the same blocks via agent i.

Ordered Partition Refinement

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** for computing bisimulation contractions.

Standard partition refinement [PT87]. Let $s = ((W, R, L), w_d)$:

- Start from an initial partition of W calculated wrt. labels.
- At each step, iterate through the elements, called **blocks**, B of the partition.
- For $x \in B$ and $x' \in B'$, we say that B **sees** B' **via agent** i **iff** xR_ix'.
- For all agents $i \in Ag$, split B into sub-blocks $B_1, \ldots, B_k$ such that the worlds in each sub-block see the same blocks via agent i.
- Iterate until no more blocks can be split: the final partition is the **set of bisimulation equivalence classes of** W.

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.
- The **signature** a world $w \in W$ wrt. a partition $(B_1, \ldots, B_k)$ is defined as follows

$$\sigma_{(B_1,\ldots,B_k)}(w) = L(w) \cup \{(i,n) \in Ag \times \mathbb{N} \mid \text{for some } v, wR_iv \text{ and } v \in B_n\}$$

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.
- The **signature** a world $w \in W$ wrt. a partition $(B_1, \ldots, B_k)$ is defined as follows

$$\sigma_{(B_1,\ldots,B_k)}(w) = L(w) \cup \{(i, n) \in Ag \times \mathbb{N} \mid \text{for some } v, wR_i v \text{ and } v \in B_n\}$$

- Signatures give **unique identifiers** of worlds wrt. a partition.

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.
- The **signature** a world $w \in W$ wrt. a partition $(B_1, \ldots, B_k)$ is defined as follows

$$\sigma_{(B_1,\ldots,B_k)}(w) = L(w) \cup \{(i,n) \in Ag \times \mathbb{N} \mid \text{for some } v, wR_i v \text{ and } v \in B_n\}$$

- Signatures give **unique identifiers** of worlds wrt. a partition.
- At each step, **blocks are split wrt. their signature**, until no more blocks can be split.

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.
- The **signature** a world $w \in W$ wrt. a partition $(B_1, \dots, B_k)$ is defined as follows

$$\sigma_{(B_1,\dots,B_k)}(w) = L(w) \cup \{(i, n) \in Ag \times \mathbb{N} \mid \text{for some } v, wR_i v \text{ and } v \in B_n\}$$

- Signatures give **unique identifiers** of worlds wrt. a partition.
- At each step, **blocks are split wrt. their signature**, until no more blocks can be split.
- The **world-set** of the contraction of s is the **set of indices of the blocks in the final partition**.

Ordered Partition Refinement (Cont.)

Bolander et al. [BDH21] propose an improved algorithm called **ordered partition refinement** (**OPT**) for computing bisimulation contractions:

- Each block B_h is given a **numerical index** h.
- The **signature** a world $w \in W$ wrt. a partition $(B_1, \ldots, B_k)$ is defined as follows

$$\sigma_{(B_1,\ldots,B_k)}(w) = L(w) \cup \{(i, n) \in Ag \times \mathbb{N} \mid \text{for some } v, wR_i v \text{ and } v \in B_n\}$$

- Signatures give **unique identifiers** of worlds wrt. a partition.
- At each step, **blocks are split wrt. their signature**, until no more blocks can be split.
- The **world-set** of the contraction of s is the **set of indices of the blocks in the final partition**.

Theorem ([BDH21])

*If $s \leftrightarrows s'$, then the contractions computed by OPT are **identical**.*

→ Bisimilarity check can be reduced to **identity check**!

Ordered Partition Refinement (Cont.)

Using ordered partition refinement, bisimilarity check can be reduced to **identity check**!

- An algorithm is provided to compute **policies** (mappings from states to actions) with a modified Pruning AND-OR (PrAO) search.
- Results show **improvements** both over a baseline planner that does not use OPT, and over the planner by Engesser et al. [Eng+17], a solver where each agent computes their policy distributively.

DEPTH-BOUNDED EPISTEMIC PLANNING

Depth-Bounded Epistemic Planning

In DEL-based epistemic planning agents can reason unboundedly about each other's knowledge.

→ This leads to **undecidability** of the plan existence problem.

→ Often unrealistic in many practical scenarios.

Depth-Bounded Epistemic Planning

In DEL-based epistemic planning agents can reason unboundedly about each other's knowledge.

- → This leads to **undecidability** of the plan existence problem.
- → Often unrealistic in many practical scenarios.

What if we **restricted the reasoning depth** of the planning agent to some bound b?

- Reduce the size of epistemic states: **bounded bisimulation contractions**.
- Look for plans requiring the lowest bound: **iterative bound-deepening search**.

Bounded Bisimulations

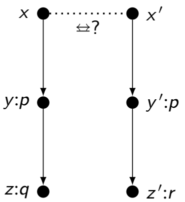
In a Nutshell: b-bisimilarity

- $x \leftrightarrows_0 x'$ iff they agree on all propositional atoms.
- $x \leftrightarrows_{b+1} x'$ iff $x \xrightarrow{i} y$ implies $x' \xrightarrow{i} y'$ and $x' \leftrightarrows_b y'$ for some y' (and vice versa).

Proposition ([BRV01])

*Two states are b-**bisimilar** iff they **satisfy the same formulas up to modal depth** b.*

Example (Are x and x' bisimilar?)



Bounded Bisimulations

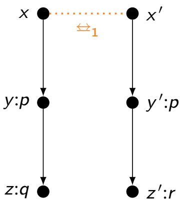
In a Nutshell: *b*-bisimilarity

- $x \leftrightarrows_0 x'$ iff they agree on all propositional atoms.
- $x \leftrightarrows_{b+1} x'$ iff $x \xrightarrow{i} y$ implies $x' \xrightarrow{i} y'$ and $x' \leftrightarrows_b y'$ for some y' (and vice versa).

Proposition ([BRV01])

Two states are ***b-bisimilar*** *iff they* ***satisfy the same formulas up to modal depth b.***

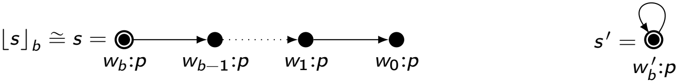
Example (Are x and x' bisimilar? No, but they are 1-bisimilar!)



Early definitions of bounded contractions in the literature did not behave as expected:

- → **Standard b-contraction**: quotient structure of a model wrt. $\leftrightarroweq_b$.
- → Standard b-contractions are in general **not minimal**.

Example (Standard (left) and minimal (right) b-contractions of a chain state)



- → Each world of the chain can be **identified** by a formula of **modal depth** $\leqslant b$.
- → Taking the quotient has no effect: we need to **keep all the worlds**.
- → **Idea**: we only need to **keep some worlds**.

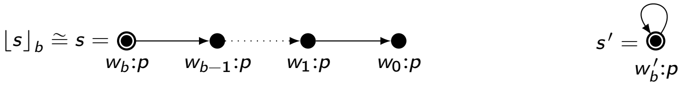
Rooted b-Contractions

Early definitions of bounded contractions in the literature did not behave as expected:

→ **Standard b-contraction**: quotient structure of a model wrt. $\leftrightarroweq_b$.

→ Standard b-contractions are in general **not minimal**.

Example (Standard (left) and minimal (right) b-contractions of a chain state)



→ Each world of the chain can be **identified** by a formula of **modal depth** $\leqslant b$.

→ Taking the quotient has no effect: we need to **keep all the worlds**.

→ **Idea**: we only need to **keep some worlds**.

$$\Downarrow$$

We improved the definition: **rooted b-contractions** guarantee **minimality**.

Canonical b-Contractions

Similar problem we had for standard contractions: **rooted b-contractions** of b-bisimilar states may be **non-isomorphic**!

$\rightarrow$ Checking for visited states is **inefficient**.

Canonical b-Contractions

Similar problem we had for standard contractions: **rooted b-contractions** of b-bisimilar states may be **non-isomorphic**!

$\rightarrow$ Checking for visited states is **inefficient**.

$\Downarrow$

Improved definition called **canonical b-contractions**, based on the notion of **h-signatures**:

$\rightarrow$ $\sigma_0(w) = (L(w), \varnothing)$

$\rightarrow$ $\sigma_{h+1}(w) = (L(w), \Sigma_{h+1}(w))$, where $\Sigma_{h+1}(w)$ maps to each agent i a set

$$\Sigma_{h+1}(w, i) = \{\sigma_h(v) \mid wR_i v\}$$

$\rightarrow$ Provide **unique identifiers** of h-bisimilar worlds.

Theorem (Identity [BBM25])

*The canonical b-contractions of b-bisimilar states are **identical**.*

From Breadth-First Search...

Let's start from a BFS with standard bisimulation contractions and check for visited states:

BFS

function BFS((s_0, Act, ϕ_g))
  $frontier \leftarrow \langle \lfloor s_0 \rfloor_{\leftrightarrow} \rangle$
  $visited \leftarrow \varnothing$
  while $\neg frontier$.empty() **do**
    $s \leftarrow frontier$.pop()
    $visited$.push(s)
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act$ applicable in s **do**
      $s' \leftarrow \lfloor s \otimes \alpha \rfloor_{\leftrightarrow}$
      If s' is not visited, push it to *frontier*
  return *fail*

Proposition ([BRV01])

Two states are ***bisimilar*** *iff they* ***satisfy the same formulas in*** $\mathcal{L}_{P,Ag}$.

Proposition ([DHK07])

If $s \leftrightarrow s'$ *and* α *is applicable in both, then* $s \otimes \alpha \leftrightarrow s' \otimes \alpha$.

…To Bounded Search

Let $\mathtt{b}_0$ be the **reasoning depth bound** of the planning agent (*i.e.*, the agent can reason to formulas with **modal depth at most** $\mathtt{b}_0$).

BoundedSearch

function BoundedSearch($(s_0, Act, \phi_g), \mathtt{b}_0$)
  $frontier \leftarrow \langle \lfloor s_0 \rfloor_{\Leftrightarrow} \rangle$
  $visited \leftarrow \varnothing$
  while $\neg frontier$.empty() **do**
    $s \leftarrow frontier$.pop()
    $visited$.push(s)
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act$ applicable in s **do**
      $s' \leftarrow \lfloor s \otimes \alpha \rfloor_{\Leftrightarrow}$
      If s' is not visited, push it to *frontier*
  return *fail*

…To Bounded Search

Let $\mathtt{b}_0$ be the **reasoning depth bound** of the planning agent (*i.e.*, the agent can reason to formulas with **modal depth at most** $\mathtt{b}_0$).

BoundedSearch

```
function BoundedSearch((s_0, Act, φ_g), b_0)
    frontier ← ⟨⟦s_0⟧*_{b_0}⟩
    visited ← ∅
    while ¬frontier.empty() do
        s ← frontier.pop()
        visited.push(s)
        if s ⊨ φ_g then return plan to s
        for all α ∈ Act applicable in s do
            s′ ← ⌊s ⊗ α⌋_↔
            If s′ is not visited, push it to frontier
    return fail
```

Proposition ([BRV01])

*Two states are **b-bisimilar** iff they **satisfy the same formulas up to modal depth** b.*

…To Bounded Search

Let $\mathtt{b}_0$ be the **reasoning depth bound** of the planning agent (*i.e.*, the agent can reason to formulas with **modal depth at most** $\mathtt{b}_0$).

BoundedSearch

function BoundedSearch$((s_0, Act, \phi_g), \mathtt{b}_0)$
  $frontier \leftarrow \langle [\![s_0]\!]^{\star}_{\mathtt{b}_0} \rangle$
  $visited \leftarrow \varnothing$
  while $\neg frontier$.empty() **do**
    $s \leftarrow frontier$.pop()
    $visited$.push(s)
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act$ applicable in s **do**
      $s' \leftarrow [\![s \otimes \alpha]\!]^{\star}_{\mathtt{b}_0}$ (?)
      If s' is not visited, push it to *frontier*
  return *fail*

...To Bounded Search

Let $\mathtt{b}_0$ be the **reasoning depth bound** of the planning agent (*i.e.*, the agent can reason to formulas with **modal depth at most** $\mathtt{b}_0$).

BoundedSearch

function BoundedSearch$((s_0, Act, \phi_g), \mathtt{b}_0)$
  $frontier \leftarrow \langle \llbracket s_0 \rrbracket^{\star}_{\mathtt{b}_0} \rangle$
  $visited \leftarrow \varnothing$
  while $\neg frontier.\mathrm{empty}()$ **do**
    $s \leftarrow frontier.\mathrm{pop}()$
    $visited.\mathrm{push}(s)$
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act$ applicable in s **do**
      $s' \leftarrow \llbracket s \otimes \alpha \rrbracket^{\star}_{\mathtt{b}_0}$ (?)
      If s' is not visited, push it to *frontier*
  return *fail*

Proposition ([BL22])

Let $s \leftrightarrows_b s'$ and let α be an action with $md(\alpha) \leqslant b$. Then, $s \otimes \alpha \leftrightarrows_{b-md(\alpha)} s' \otimes \alpha$.

Where $md(\alpha)$ denotes the **maximal modal depth** of all pre- and postconditions in α.

...To Bounded Search

Let $\mathtt{b}_0$ be the **reasoning depth bound** of the planning agent (*i.e.*, the agent can reason to formulas with **modal depth at most** $\mathtt{b}_0$).

BoundedSearch

```
function BoundedSearch((s_0, Act, φ_g), b_0)
    frontier ← ⟨⟦s_0⟧*_{b_0}⟩
    visited ← ∅
    while ¬frontier.empty() do
        s ← frontier.pop()
        visited.push(s)
        if s ⊨ φ_g then return plan to s
        for all α ∈ Act applicable in s do
            ~~s' ← ⟦s ⊗ α⟧*_{b_0}~~
            If s' is not visited, push it to frontier
    return fail
```

Proposition ([BL22])

Let $s \leftrightarrows_b s'$ *and let* α *be an action with* $md(\alpha) \leqslant b$. *Then,* $s \otimes \alpha \leftrightarrows_{b-md(\alpha)} s' \otimes \alpha$.

Where $md(\alpha)$ denotes the **maximal modal depth** of all pre- and postconditions in α.

→ We need to **update** the bound value after an update.

Updating Bounds Value After Updates

We let a **node** of the search space be a pair $n = (s, b)$, where:

1. s is the **state** of n (denoted n.state).
2. b is the **(depth) bound** (denoted n.bound) $\rightarrow$ **maximum modal depth** of formulas we can safely evaluate in s.

Updating Bounds Value After Updates

We let a **node** of the search space be a pair $n = (s, b)$, where:

1. s is the **state** of n (denoted n.state).
2. b is the **(depth) bound** (denoted n.bound) $\rightarrow$ **maximum modal depth** of formulas we can safely evaluate in s.

In general, s will be a b-**contracted state** that can be thought of as an **approximation to the "real" state**.

$\rightarrow$ We are always guaranteed that s is **at least b-bisimilar to the real state**.

Putting Everything Together

BoundedSearch

```
function BoundedSearch((s_0, Act, φ_g), b_0)
    frontier ← ⟨(⟦s_0⟧*_{b_0}, b_0)⟩
    visited ← ∅
    while ¬frontier.empty() do
        (s, b) ← frontier.pop()
        visited.push(s)
        if s ⊨ φ_g then return plan to s
        for all α ∈ Act | b ⩾ md(α) do
            if α is applicable in s then
                s′ ← ⟦s ⊗ α⟧*_{b−md(α)}
                n′ ← (s′, b − md(α))
                If s′ is not visited, push n′ to frontier
    return fail
```

Proposition ([BRV01])

*Two states are **b-bisimilar** iff they satisfy **the same formulas up to modal depth** b.*

Proposition ([BL22])

Let $s \leftrightarroweq_b s'$ and let α be an action with $md(\alpha) \leqslant b$. Then, $s \otimes \alpha \leftrightarroweq_{b-md(\alpha)} s' \otimes \alpha$.

Putting Everything Together

BoundedSearch

function BoundedSearch($(s_0, Act, \phi_g), b_0$)
  frontier $\leftarrow \langle(\llbracket s_0 \rrbracket^{\star}_{b_0}, b_0)\rangle$
  visited $\leftarrow \varnothing$
  while $\neg$*frontier*.empty() **do**
    $(s, b) \leftarrow$ *frontier*.pop()
    visited.push(s)
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act \mid b \geqslant md(\alpha) + md(\phi_g)$ **do**
      if α is applicable in s **then**
        $s' \leftarrow \llbracket s \otimes \alpha \rrbracket^{\star}_{b - md(\alpha)}$
        $n' \leftarrow (s', b - md(\alpha))$
        If s' is not visited, push n' to *frontier*
  return *fail*

Proposition ([BRV01])

*Two states are **b-bisimilar** iff they **satisfy the same formulas up to modal depth** b.*

Proposition ([BL22])

Let $s \leftrightarrows_b s'$ and let α be an action with $md(\alpha) \leqslant b$. Then, $s \otimes \alpha \leftrightarrows_{b-md(\alpha)} s' \otimes \alpha$.

Putting Everything Together

BoundedSearch

function BoundedSearch$((s_0, Act, \phi_g), b_0)$
  $frontier \leftarrow \langle (\llbracket s_0 \rrbracket^{\star}_{b_0}, b_0) \rangle$
  $visited \leftarrow \varnothing$
  while $\neg frontier$.empty() **do**
    $(s, b) \leftarrow frontier$.pop()
    $visited$.push(s)
    if $s \models \phi_g$ **then return** plan to s
    for all $\alpha \in Act \mid b \geqslant md(\alpha) + md(\phi_g)$ **do**
      if α is applicable in s **then**
        $s' \leftarrow \llbracket s \otimes \alpha \rrbracket^{\star}_{b-md(\alpha)}$
        $n' \leftarrow (s', b - md(\alpha))$
        if $s' \notin visited$ **then** $frontier$.push(n')
  return *fail*

Proposition ([BRV01])

*Two states are **b-bisimilar** iff they **satisfy the same formulas up to modal depth** b.*

Proposition ([BL22])

Let $s \leftrightarrows_b s'$ and let α be an action with $md(\alpha) \leqslant b$. Then, $s \otimes \alpha \leftrightarrows_{b-md(\alpha)} s' \otimes \alpha$.

Theorem ([BBM25])

*The canonical b-contractions of b-bisimilar states are **identical**.*

Iterative Bound-Deepening Search

function IBDS($T = (s_0, Act, \phi_g)$)
  for $b \leftarrow md(\phi_g)$ **to** ∞ **do**
    $\pi \leftarrow$ BoundedSearch(T, b)
    if $\pi \neq fail$ **then return** π

We call **BoundedSearch** over increasing values of b:

$\rightarrow$ If $b < md(\phi_g)$, then the **bound is too low** to safely evaluate the goal formula.

$\rightarrow$ So initially we let $b = md(\phi_g)$.

$\rightarrow$ If no goal is found with bound b, we **increment the bound and try again**.

Improving Bounded Search

In a node $n = (s, b)$, the state s can be considered as an **approximation to modal depth** b of some "true state" t (namely, we are guaranteed that $s \leftrightarroweq_b t$). However:

Improving Bounded Search

In a node $n = (s, b)$, the state s can be considered as an **approximation to modal depth** b of some "true state" t (namely, we are guaranteed that $s \leftrightarroweq_b t$). However:

- In general it could be that $s \not\leftrightarroweq t$!

Improving Bounded Search

In a node $n = (s, b)$, the state s can be considered as an **approximation to modal depth b** of some "true state" t (namely, we are guaranteed that $s \leftrightarroweq_b t$). However:

- In general it could be that $s \leftrightarroweq t$!
- In this case, when we update s with an action α, we don't have to decrease the bound.
  - $\rightarrow$ Recall that **bisimilarity is preserved** after product update!

Improving Bounded Search

In a node $n = (s, b)$, the state s can be considered as an **approximation to modal depth** b of some "true state" t (namely, we are guaranteed that $s \leftrightarroweq_b t$). However:

- In general it could be that $s \leftrightarroweq t$!
- In this case, when we update s with an action α, we don't have to decrease the bound.
  - $\rightarrow$ Recall that **bisimilarity is preserved** after product update!

We can use this idea to include the following **optimizations** in BoundedSearch:

- We add a **third parameter** called *is_bisim* to our nodes, representing **whether the state of a node is bisimilar to its corresponding true state**.
- Depending on whether *is_bisim* holds, we **update a node with the appropriate bound value**.

Improving Bounded Search

In a node $n = (s, b)$, the state s can be considered as an **approximation to modal depth** b of some "true state" t (namely, we are guaranteed that $s \leftrightarroweq_b t$). However:

- In general it could be that $s \leftrightarroweq t$!
- In this case, when we update s with an action α, we don't have to decrease the bound.
  - $\rightarrow$ Recall that **bisimilarity is preserved** after product update!

We can use this idea to include the following **optimizations** in BoundedSearch:

- We add a **third parameter** called *is_bisim* to our nodes, representing **whether the state of a node is bisimilar to its corresponding true state**.
- Depending on whether *is_bisim* holds, we **update a node with the appropriate bound value**.
- Across different iterations of IBDS, we **preserve all nodes having *is_bisim* true**.
  - $\rightarrow$ They would otherwise be **recomputed** in the next iteration!

Soundness, Completeness, Complexity

Let $T = (s_0, Act, \phi_g)$ be a planning task and let $b \geqslant md(\phi_g)$ be a constant.

Theorem (Soundness)

If **BoundedSearch**(T, b) *returns an action sequence* π*, then* π *is a solution to* T*.*

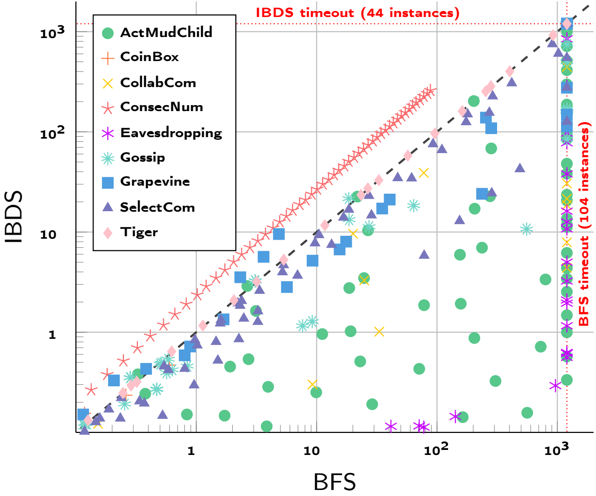
Theorem (Completeness)

If T *has a solution of length* ℓ*, then* **BoundedSearch**$(T, c \cdot \ell + md(\phi_g))$ *will find a solution to it, where* $c = \max\{md(\alpha) \mid \alpha \in Act\}$*.*

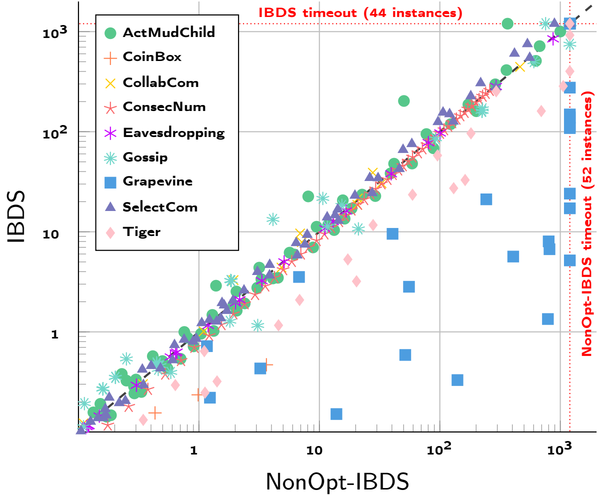
Theorem (Complexity)

BoundedSearch *runs in* $(b{+}1)$-***ExpTime***.

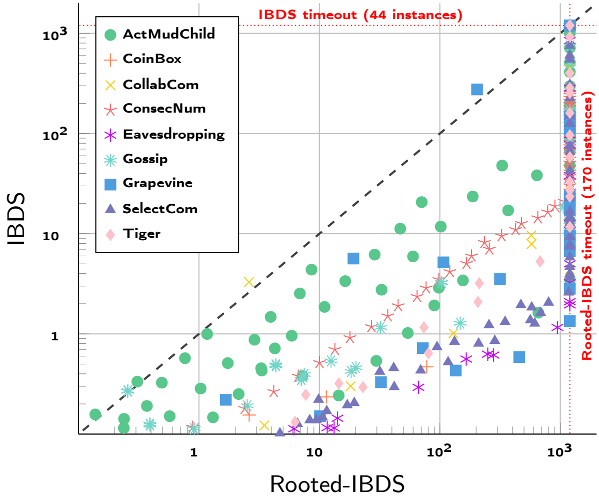
IBDS vs. BFS


IBDS timeout (44 instances)
BFS timeout (104 instances)
ActMudChild
CoinBox
CollabCom
ConsecNum
Eavesdropping
Gossip
Grapevine
SelectCom
Tiger
IBDS
BFS
1
10
10^2
10^3

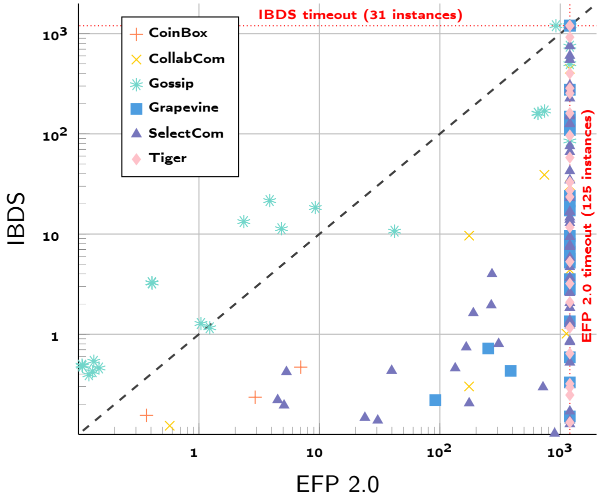
IBDS vs. Non-Optimized IBDS



Canonical vs. Rooted Contractions



IBDS vs. EFP 2.0



FUTURE DIRECTIONS

Many Ideas to Try Out

- DEL-based epistemic planning is a **hard problem**.
- Despite this, there have been **many recent promising advancements**.
- **Different ideas** have been explored, from compilation-based techniques, to heuristics, to bisimulation contractions.
- Many ideas haven't been tried yet!
  - → **Symbolic** approaches, **SAT/SMT-based** epistemic planning, more heuristics.

One Language to Compare Them All

So many different frameworks, with many different semantics. How can we compare them?

- The **Epistemic Planning Domain Definition Language**.
- Combines a PDDL-like syntax with the full DEL semantics.
- **Different formalisms/fragments** can be define within the **same language**!
- Will soon be released!

Exciting News!

- See you all **next year** in Dublin for the **first Epistemic Planning Track** at the IPC!
- More details will be given at the **ICAPS Community Meeting** (**Thursday, November 13, at 15:30**).

Exciting News!

- See you all **next year** in Dublin for the **first Epistemic Planning Track** at the IPC!
- More details will be given at the **ICAPS Community Meeting** (**Thursday, November 13, at 15:30**).

Thank you! Questions?