Algorithms and Satisfiability

. Mini-Project: Planning

Solving Problems using Planning

Álvaro Torralba

AALBORG UNIVERSITET

Spring 2023

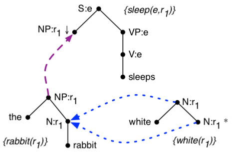
Thanks to Jörg Hoffmann for slide sources

Agenda

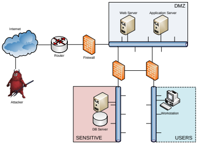
1. Introduction
2. The PDDL Language
3. Beyond Classical Planning
4. Conclusion

Agenda

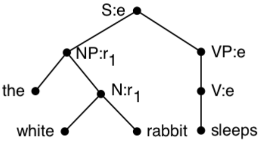
Planning

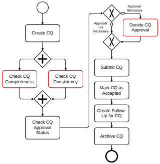


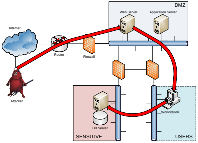
Action name	precondition	effect
Check CQ Completeness	CQ.archiving:notArchived	CQ.completeness:complete OR CQ.completeness:notComplete
Check CQ Consistency	CQ.archiving:notArchived	CQ.consistency:consistent OR CQ.consistency:notConsistent
Check CQ Approval Status	CQ.archiving:notArchived AND CQ.approval:notChecked AND CQ.completeness:complete AND CQ.consistency:consistent	CQ.approval:necessary OR CQ.approval:notNecessary
Decide CQ Approval	CQ.archiving:notArchived AND CQ.approval:necessary	CQ.approval:granted OR CQ.approval:notGranted
Submit CQ	CQ.archiving:notArchived AND (CQ.approval:notNecessary OR CQ.approval:granted)	CQ.submission:submitted
Mark CQ as Accepted	CQ.archiving:notArchived AND CQ.submission:submitted	CQ.acceptance:accepted
Create Follow-Up for CQ	CQ.archiving:notArchived AND CQ.acceptance:accepted	CQ.followUp:documentCreated
Archive CQ	CQ.archiving:notArchived	CQ.archiving:archived



Planning Domain Definition Language (PDDL) $\mapsto$ Planning System







STRIPS Planning: Syntax

Definition (STRIPS Planning Task). *A STRIPS planning task, short planning task, is a 4-tuple $\Pi = (P, A, I, G)$ where:*

- *P is a finite set of facts (aka propositions).*
- *A is a finite set of actions; each $a \in A$ is a triple $a = (pre_a, add_a, del_a)$ of subsets of P referred to as the action's precondition, add list, and delete list respectively; we require that $add_a \cap del_a = \emptyset$.*
- *$I \subseteq P$ is the initial state.*
- *$G \subseteq P$ is the goal.*

We will often give each action $a \in A$ a name (a string), and identify a with that name.

STRIPS Planning: Syntax

Definition (STRIPS Planning Task). *A STRIPS planning task, short planning task, is a 4-tuple $\Pi = (P, A, I, G)$ where:*

- *P is a finite set of facts (aka propositions).*
- *A is a finite set of actions; each $a \in A$ is a triple $a = (pre_a, add_a, del_a)$ of subsets of P referred to as the action's precondition, add list, and delete list respectively; we require that $add_a \cap del_a = \emptyset$.*
- *$I \subseteq P$ is the initial state.*
- *$G \subseteq P$ is the goal.*

We will often give each action $a \in A$ a name (a string), and identify a with that name.

Note: We assume unit costs for simplicity: every action has cost 1.

Agenda

1. Introduction
2. The PDDL Language
3. Beyond Classical Planning
4. Conclusion

PDDL History

Planning Domain Description Language:

- A description language for planning in the STRIPS formalism and various extensions.
- Used in the International Planning Competition (IPC).

- 1998: PDDL [McDermott *et al.* (1998)].
- 2000: "PDDL subset for the 2000 competition" [Bacchus (2000)].
- 2002: PDDL2.1, Levels 1-3 [Fox and Long (2003)].
- 2004: PDDL2.2 [Hoffmann and Edelkamp (2005)].
- 2006: PDDL3 [Gerevini *et al.* (2009)].

PDDL Quick Facts

PDDL is not a propositional language:

- Representation is lifted, using object variables to be instantiated from a finite set of objects. (Similar to predicate logic)
- Action schemas parameterized by objects.
- Predicates to be instantiated with objects.

PDDL Quick Facts

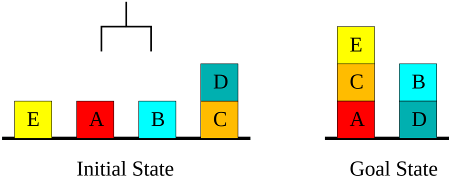
PDDL is not a propositional language:

- Representation is lifted, using object variables to be instantiated from a finite set of objects. (Similar to predicate logic)
- Action schemas parameterized by objects.
- Predicates to be instantiated with objects.

A PDDL planning task comes in two pieces:

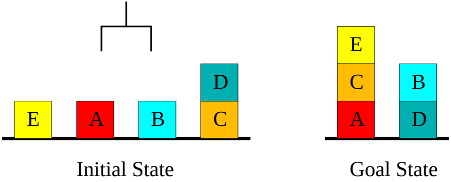
- The domain file and the problem file.
- The problem file gives the objects, the initial state, and the goal state.
- The domain file gives the predicates and the action schemas; each benchmark domain has *one* domain file.

The Blocksworld in PDDL (STRIPS): Domain File




```
(define (domain blocksworld)
(:predicates (clear ?x) (holding ?x) (on ?x ?y)
             (on-table ?x) (arm-empty))
(:action stack
 :parameters (?x ?y)
 :precondition (and (clear ?y) (holding ?x))
 :effect (and (arm-empty) (on ?x ?y)
              (not (clear ?y)) (not (holding ?x)))
)
...
```

The Blocksworld in PDDL (STRIPS): Problem File




```
(define (problem bw-abcde)
(:domain blocksworld)
(:objects a b c d e)
(:init (on-table a) (clear a)
       (on-table b) (clear b)
       (on-table e) (clear e)
       (on-table c) (on d c) (clear d)
       (arm-empty))
(:goal (and (on e c) (on c a) (on b d))))
```

Fast Downward

Fast Downward is a planning system featuring a lot of algorithms. When you run it you need to select which configuration to use:

```
./fast-downward.py (<domain>) <instance> --search "config"
```

```
./fast-downward.py --alias "config-alias" (<domain>) <instance>
```

There are A LOT of configurations. Here I list a few convenient ones:

Satisficing Planning:

- What we see in the lecture:
  `--evaluator "hff=ff(transform=adapt_costs(one))" --search "eager_greedy([hff], preferred=[hff], cost_type=one)"`
- `--alias lama-first`: Good configuration
- `--alias lama`: Good configuration (anytime)

Optimal Planning:

- `--search "astar(blind)"`: Dijkstra search
- `--search "astar(lmcut)"`: Ok configuration (though not best)

Action Description Language (ADL)

STRIPS + ADL (Action Description Language):

- Arbitrary first-order logic formulas in action preconditions and the goal: forall, exists, or, imply, not
- Conditional effects, i.e., effects that occur only if their separate effect condition holds: when

→A useful construct is effects of the form forall-when:

```
(forall (?x) (when (condition) (effect))
```

ADL is a real headache to implement:

- Most planners that do handle ADL *compile* it down [Gazen and Knoblock (1997)]
- Example FF: 7000 C lines for compilation, 2000 lines core planner.

Action Costs

`(:requirements :action-costs)`

Domain file:

- Declare cost function

```
(:functions
    (road-length ?l1 ?l2 - location) - number ; optional
    (total-cost) - number ; The cost function must have this name
)
```

- Declare action cost as effect:

  `(increase (total-cost) (road-length ?l1 ?l2))`

Problem file:

- (optional) Declare costs in the initial state:

  `(= (total-cost) 0)`

  `(= (road-length city-3-loc-2 city-2-loc-3) 186)`

- Optimization criteria: `(:metric minimize (total-cost))`

PDDL Extensions

- PDDL 2.1: numeric and temporal planning
- PDDL 2.2: derived predicates (e.g., flow of current in an electricity network) and timed initial literals (e.g., sunrise and sunset, shop closing times).
- PDDL 3: soft goals (e.g.goals that have a reward) and preferences (e.g.temporal goals)

PDDL Extensions

- PDDL 2.1: numeric and temporal planning
- PDDL 2.2: derived predicates (e.g., flow of current in an electricity network) and timed initial literals (e.g., sunrise and sunset, shop closing times).
- PDDL 3: soft goals (e.g.goals that have a reward) and preferences (e.g.temporal goals)

In practice, most planners only support a subset of PDDL. In this project, you should consider:

- STRIPS
- Negative Preconditions
- Forall-when effects
- Action costs

Questionnaire

Question!

What is PDDL good for?

(A): Nothing.

(B): Free beer.

(C): Those AI planning guys.

(D): Being lazy at work.

Questionnaire

Question!

What is PDDL good for?

(A): Nothing.

(B): Free beer.

(C): Those AI planning guys.

(D): Being lazy at work.

→ (A): Nah, it's definitely good for *something* (see remaining answers).

Questionnaire

Question!

What is PDDL good for?

(A): Nothing. (B): Free beer.

(C): Those AI planning guys. (D): Being lazy at work.

→ (A): Nah, it's definitely good for *something* (see remaining answers).

→ (B): Generally, no. Sometimes, yes: PDDL is needed for the IPC, and if you win the IPC you get price money (= free beer).

Questionnaire

Question!

What is PDDL good for?

(A): Nothing. (B): Free beer.

(C): Those AI planning guys. (D): Being lazy at work.

→ (A): Nah, it's definitely good for *something* (see remaining answers).

→ (B): Generally, no. Sometimes, yes: PDDL is needed for the IPC, and if you win the IPC you get price money (= free beer).

→ (C): Yep. (When I started in this area, every system had its own language, so running experiments felt a lot like "Lost in Translation".)

Questionnaire

Question!

What is PDDL good for?

(A): Nothing.

(B): Free beer.

(C): Those AI planning guys.

(D): Being lazy at work.

→ (A): Nah, it's definitely good for *something* (see remaining answers).

→ (B): Generally, no. Sometimes, yes: PDDL is needed for the IPC, and if you win the IPC you get price money (= free beer).

→ (C): Yep. (When I started in this area, every system had its own language, so running experiments felt a lot like "Lost in Translation".)

→ (D): Yep. You can be a busy bee, programming a solver yourself. Or you can be lazy and just write the PDDL. (I think I said that before . . .)

Agenda

Beyond Classical Planning

Classical Planning models sequential-decision making under some assumptions over the environment:

Beyond Classical Planning

Classical Planning models sequential-decision making under some assumptions over the environment:

Fully observable, Deterministic, Static, Discrete, Single agent

Beyond Classical Planning

Classical Planning models sequential-decision making under some assumptions over the environment:

Fully observable, Deterministic, Static, Discrete, Single agent

There are many extensions that go beyond this:

- **Planning with Uncertainty:** partially-observable environments.
- **Non-deterministic Planning:** non-deterministic environments.
- **Numeric Planning:** continous environments.
- **Multi-agent Planning:** environments where several agents cooperate.
- **Temporal Planning:** actions whose effects take some time.
- And more...

Planning with Uncertainty

In the real world: we do not know everything!

We have only partial knowledge about the initial state.
How to represent our knowledge?

Planning with Uncertainty

In the real world: we do not know everything!

We have only partial knowledge about the initial state.

How to represent our knowledge? → Logic!

Planning with Uncertainty

In the real world: we do not know everything!

We have only partial knowledge about the initial state.
How to represent our knowledge? $\rightarrow$ Logic!

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 $\boxed{A}$ OK	2,1 OK	3,1	4,1

Initial knowledge:

- You're in cell [1,1]. $P_{1,1}$
- There's a Wumpus $(W_{1,1} \vee W_{1,2} \vee W_{1,3} \vee \dots)$
- There's gold $(G_{1,1} \vee G_{1,2} \vee G_{1,3} \vee \dots)$
- There's no stench in position [1, 1]: $\neg S_{1,1}$
- General knowledge:
  $\neg S_{1,1} \rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

Planning with Uncertainty

In the real world: we do not know everything!

We have only partial knowledge about the initial state.
How to represent our knowledge? $\rightarrow$ Logic!

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 $\boxed{A}$ OK	2,1 OK	3,1	4,1

Initial knowledge:

- You're in cell [1,1]. $P_{1,1}$
- There's a Wumpus $(W_{1,1} \vee W_{1,2} \vee W_{1,3} \vee \dots)$
- There's gold $(G_{1,1} \vee G_{1,2} \vee G_{1,3} \vee \dots)$
- There's no stench in position [1, 1]: $\neg S_{1,1}$
- General knowledge: $\neg S_{1,1} \rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

Definition (Belief State). *Let φ be a propositional formula that describes our knowledge about the current state. Then, the belief state B is the set of states that correspond to satisfying assignments of φ.*

$\rightarrow$The set of states that are consistent with our belief.

Partially-Observable Planning

A planning task with uncertainty in the initial state, is a 4-tuple $\Pi = (P, A, \varphi_I, G)$ *where:*

- P is a finite set of facts.
- A is a finite set of actions; each $a \in A$ is a tuple (pre, add, del, obs). $\rightarrow$The value of facts in **obs** set after executing the action.
- φ_I is the initial belief state.
- $G \subseteq P$ is the goal.

Partially-Observable Planning

A planning task with uncertainty in the initial state, is a 4-tuple $\Pi = (P, A, \varphi_I, G)$ *where:*

- P is a finite set of facts.
- A is a finite set of actions; each $a \in A$ is a tuple (pre, add, del, obs). $\rightarrow$The value of facts in **obs** set after executing the action.
- φ_I is the initial belief state.
- $G \subseteq P$ is the goal.

$\rightarrow$**Conformant Planning:** Find a sequence of actions that transform the initial belief state φ_I into $\varphi_G \models G$ (conformant plan). Our plan works no matter in which initial state we are. [EXPSPACE-hard]

Partially-Observable Planning

A planning task with uncertainty in the initial state, is a 4-tuple $\Pi = (P, A, \varphi_I, G)$ *where:*

- P is a finite set of facts.
- A is a finite set of actions; each $a \in A$ is a tuple (pre, add, del, obs). $\rightarrow$The value of facts in **obs** set after executing the action.
- φ_I is the initial belief state.
- $G \subseteq P$ is the goal.

$\rightarrow$**Conformant Planning:** Find a sequence of actions that transform the initial belief state φ_I into $\varphi_G \models G$ (conformant plan). Our plan works no matter in which initial state we are. [EXPSPACE-hard]

$\rightarrow$**Contingent Planning:** Find a tree of actions that transform the initial belief state φ_I into $\varphi_G \models G$. We may need different plans for every result of the observation actions! [2EXPSPACE-hard]

Non-deterministic/Stochastic Environments

In the real world: we cannot always anticipate the effect of our actions!

Differences with respect to classical planning:

Non-deterministic/Stochastic Environments

In the real world: we cannot always anticipate the effect of our actions!

Differences with respect to classical planning:

1. Actions can have multiple outcomes
2. (Optionally) If the probability of each outcome is known, then we call it probabilistic planning.

Non-deterministic/Stochastic Environments

In the real world: we cannot always anticipate the effect of our actions!

Differences with respect to classical planning:

1. Actions can have multiple outcomes
2. (Optionally) If the probability of each outcome is known, then we call it probabilistic planning.

Extensions of PDDL:

- PPDDL
- RDDL

Online vs. Offline Planning

Given a non-deterministic/probabilistic planning task:
Offline Planning: Plan ahead for every possibility.

- Find contingent plan
- Find (optimal) policy

Online vs. Offline Planning

Given a non-deterministic/probabilistic planning task:
Offline Planning: Plan ahead for every possibility.

- Find contingent plan
- Find (optimal) policy

Online Planning: Decide what to do next: spent some time deciding what action to execute, execute it, observe the result and re-plan if necessary.

Online vs. Offline Planning

Given a non-deterministic/probabilistic planning task:

Offline Planning: Plan ahead for every possibility.

- Find contingent plan
- Find (optimal) policy

Online Planning: Decide what to do next: spent some time deciding what action to execute, execute it, observe the result and re-plan if necessary.

FF-Replan. Given a probabilistic planning task.

1. Drop probabilities away (assume that you get to choose the outcome)
2. Use a classical problem to solve the simplified task
3. Execute the action recommended by the planner
4. If the outcome is the expected one, continue. Otherwise, re-plan from the state.

Online vs. Offline Planning

Given a non-deterministic/probabilistic planning task:
Offline Planning: Plan ahead for every possibility.

- Find contingent plan
- Find (optimal) policy

Online Planning: Decide what to do next: spent some time deciding what action to execute, execute it, observe the result and re-plan if necessary.

FF-Replan. Given a probabilistic planning task.

1. Drop probabilities away (assume that you get to choose the outcome)
2. Use a classical problem to solve the simplified task
3. Execute the action recommended by the planner
4. If the outcome is the expected one, continue. Otherwise, re-plan from the state.

$\rightarrow$Very effective online probabilistic planner, specially in tasks where probabilities model that actions have a (low) probability of failure.

Numeric Planning

In the real world: We have numbers!

Numeric STRIPS Planning: Extends STRIPS by introducing numeric variables V_n with rational values in $\mathbb{Q}$.

- Numeric expressions: We can do simple arithmetic $(+, -, \times, \div)$ with the values of variables and/or constants.

Numeric Planning

In the real world: We have numbers!

Numeric STRIPS Planning: Extends STRIPS by introducing numeric variables V_n with rational values in $\mathbb{Q}$.

- Numeric expressions: We can do simple arithmetic $(+, -, \times, \div)$ with the values of variables and/or constants.
- Numeric conditions: compare numeric expressions with $\{<, \leq, =, \geq, >\}$.

Numeric Planning

In the real world: We have numbers!

Numeric STRIPS Planning: Extends STRIPS by introducing numeric variables V_n with rational values in $\mathbb{Q}$.

- Numeric expressions: We can do simple arithmetic $(+, -, \times, \div)$ with the values of variables and/or constants.
- Numeric conditions: compare numeric expressions with $\{<, \leq, =, \geq, >\}$.
- Numeric effects: assign a numeric expression to a numeric variable in V_n.

Example: drive (x,y):

$pre : \{at(x), fuel > 0\}, add : \{at(y)\} del : \{at(x)\},$
$assign : \{fuel := fuel - 1\}$

Numeric Planning

In the real world: We have numbers!

Numeric STRIPS Planning: Extends STRIPS by introducing numeric variables V_n with rational values in $\mathbb{Q}$.

- Numeric expressions: We can do simple arithmetic $(+, -, \times, \div)$ with the values of variables and/or constants.
- Numeric conditions: compare numeric expressions with $\{<, \leq, =, \geq, >\}$.
- Numeric effects: assign a numeric expression to a numeric variable in V_n.

Example: drive (x,y):
$pre : \{at(x), fuel > 0\}, add : \{at(y)\} del : \{at(x)\},$
$assign : \{fuel := fuel - 1\}$

[Undecidable]

Multi-Agent Planning

In the real world: We are not the single and only agent!

Multi-agent planning: Several agents must collaborate to achieve a common goal.
Key: There is some global information known by all agents but each agent has his own private facts, who do not want to share with the rest.

Multi-Agent Planning

In the real world: We are not the single and only agent!

Multi-agent planning: Several agents must collaborate to achieve a common goal.
Key: There is some global information known by all agents but each agent has his own private facts, who do not want to share with the rest.

Multi-Agent STRIPS Planning: A multi-agent STRIPS planning task, is a 5-tuple $\Pi = (P, A, I, G)$ where:

- P is a finite set of facts, divided in private and public facts.
- A is a finite set of actions; each $a \in A$ is a triple of pre, add, and del, divided in private and public actions.
- $I \subseteq P$ is the initial state.
- $G \subseteq P$ is the goal.

Multi-Agent Planning

In the real world: We are not the single and only agent!

Multi-agent planning: Several agents must collaborate to achieve a common goal.
Key: There is some global information known by all agents but each agent has his own private facts, who do not want to share with the rest.

Multi-Agent STRIPS Planning: A multi-agent STRIPS planning task, is a 5-tuple $\Pi = (P, A, I, G)$ where:

- P is a finite set of facts, divided in private and public facts.
- A is a finite set of actions; each $a \in A$ is a triple of pre, add, and del, divided in private and public actions.
- $I \subseteq P$ is the initial state.
- $G \subseteq P$ is the goal.

→Agents must communicate during the planning process to share information about how they will achieve the goal

Temporal Planning (PDDL 2.2)

In the real world: Events do not happen instantaneously!

Temporal Planning (PDDL 2.2)

In the real world: Events do not happen instantaneously!

In classical planning the action effects happen immediately. However, in the real world, actions take time to execute.
When the precondition needs to hold? When the effect is applied?

Temporal Planning (PDDL 2.2)

In the real world: Events do not happen instantaneously!

In classical planning the action effects happen immediately. However, in the real world, actions take time to execute.

When the precondition needs to hold? When the effect is applied?

- preconditions at-start, and effects at-end

Temporal Planning (PDDL 2.2)

In the real world: Events do not happen instantaneously!

In classical planning the action effects happen immediately. However, in the real world, actions take time to execute.

When the precondition needs to hold? When the effect is applied?

- preconditions at-start, and effects at-end
- preconditions at-end
- effects at-end

Temporal Planning (PDDL 2.2)

In the real world: Events do not happen instantaneously!

In classical planning the action effects happen immediately. However, in the real world, actions take time to execute.
When the precondition needs to hold? When the effect is applied?

- preconditions at-start, and effects at-end
- preconditions at-end
- effects at-end
- over-all: during the execution of the action

Planning in the Real World

Question!

If most real-world environments are not deterministic, not fully observable, not discrete, not single agent, and temporal. What is classical planning good for?

Planning in the Real World

Question!

If most real-world environments are not deterministic, not fully observable, not discrete, not single agent, and temporal. What is classical planning good for?

- The model does not try to simulate the environment, it is just a tool to take good decisions.

Planning in the Real World

Question!

If most real-world environments are not deterministic, not fully observable, not discrete, not single agent, and temporal. What is classical planning good for?

- The model does not try to simulate the environment, it is just a tool to take good decisions.
- Oftentimes, reasoning with a simplified model can still lead to intelligent decisions and solutions are easier to compute than with more complex models.
- Classical planning is a relaxation of the problem so it can be used in heuristics for more complex types of planning.

Planning in the Real World

Question!

If most real-world environments are not deterministic, not fully observable, not discrete, not single agent, and temporal. What is classical planning good for?

- The model does not try to simulate the environment, it is just a tool to take good decisions.
- Oftentimes, reasoning with a simplified model can still lead to intelligent decisions and solutions are easier to compute than with more complex models.
- Classical planning is a relaxation of the problem so it can be used in heuristics for more complex types of planning.

My two cents: Ideally, we should always provide an accurate description of the environment so that the AI simplifies it when necessary. However, automatic simplification methods are not powerful enough in all cases yet.

Agenda

1. Introduction
2. The PDDL Language
3. Beyond Classical Planning
4. Conclusion

Summary

- You can use planning tools to solve your problems:
  →Encode them in PDDL and use any planner
- Classical planning is very effective at solving large problems
- Non-classical planning models reason about more complex environments such as non-deterministic, partially-observable, continuous, temporal, etc.
- Solving these problems by computing a complete offline policy is hard (though many non-classical planners are able to do this satisfactorily in some domains). Many approaches are online, planning to decide the next action by looking into the future but without considering all alternatives.
- Classical planning and heuristic search techniques are still an important ingredient of many approaches that deal with complex environments.

Additional Resources

- Editor: `http://editor.planning.domains/`
- VS Extension: https://marketplace.visualstudio.com/items?itemName=jan-dolejsi.pddl
- Wiki: `https://planning.wiki/`
- Benchmarks: `https://github.com/aibasel/downward-benchmarks`
- Planners:
  - Fast Downward: `http://www.fast-downward.org/`
  - IPC-18: `https://ipc2018-classical.bitbucket.io/#planners`
- Domains as examples:
  - STRIPS: FD'All → Blocksworld, Logistics
  - Action costs: IPC'11 → Woodworking, Transport
  - Discretized numbers: IPC'11 → Nomystery (fuel consumption)
  - Forall-when: IPC'18 → Nurikabe
  - Complex ADL: FD'All → Miconic-ADL (ID114)

References I

Fahiem Bacchus. *Subset of PDDL for the AIPS2000 Planning Competition*. The AIPS-00 Planning Competition Comitee, 2000.

Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

B. Cenk Gazen and Craig Knoblock. Combining the expressiveness of UCPOP with the efficiency of Graphplan. In S. Steel and R. Alami, editors, *Proceedings of the 4th European Conference on Planning (ECP'97)*, pages 221–233. Springer-Verlag, 1997.

Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

Jörg Hoffmann and Stefan Edelkamp. The deterministic part of ipc-4: An overview. *Journal of Artificial Intelligence Research*, 24:519–579, 2005.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. *The PDDL Planning Domain Definition Language*. The AIPS-98 Planning Competition Comitee, 1998.