Stavros Vassos, University of Athens, Greece stavrosv@di.uoa.gr May 2012

INTRODUCTION TO AI STRIPS PLANNING

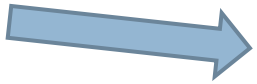
.. and Applications to Video-games!

Course overview

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

STRIPS planning

- Given:
  - Initial state

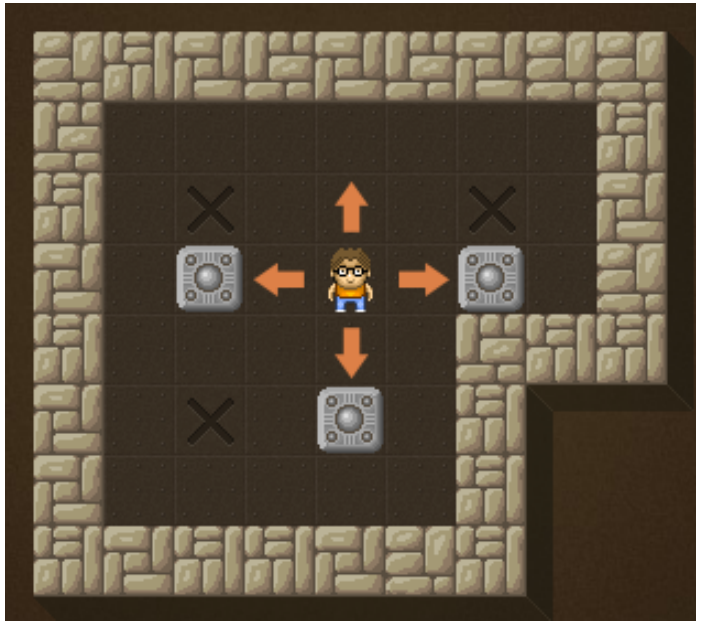


STRIPS planning

- Given:
  - Initial state
  - Goal

STRIPS planning

- Given:
  - Initial state
  - Goal
  - Available actions



STRIPS planning

- Given:
  - Initial state
  - Goal
  - Available actions

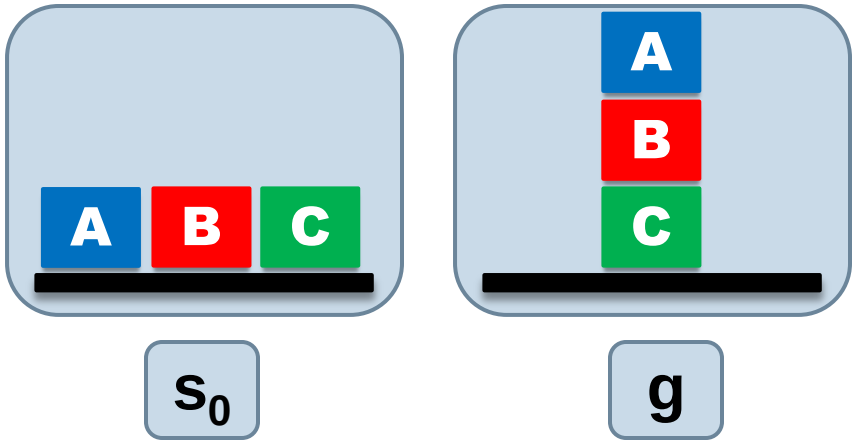
- Find:
  - A **sequence of actions** that achieves the goal
  - E.g.: **[Left, Down, Left, Up, …]**

Planning Domain Description Language

- Planning Domain Definition Language (PDDL)
  - Formal language for specifying **planning problems**
  - Formal syntax similar to a **programming language**
  - Includes **STRIPS** and ADL, and many more features
  - Provides the ground for performing a direct comparison between planning techniques and evaluating against classes of problems

Planning Domain Description Language

- Blocks world domain
  - Initial state: s_0
  - Goal: g



  - Available actions: moving a block
    - from the table to the top of another block
    - from the top of another block to the table
    - from the top of one block to the top of another block

Planning Domain Description Language

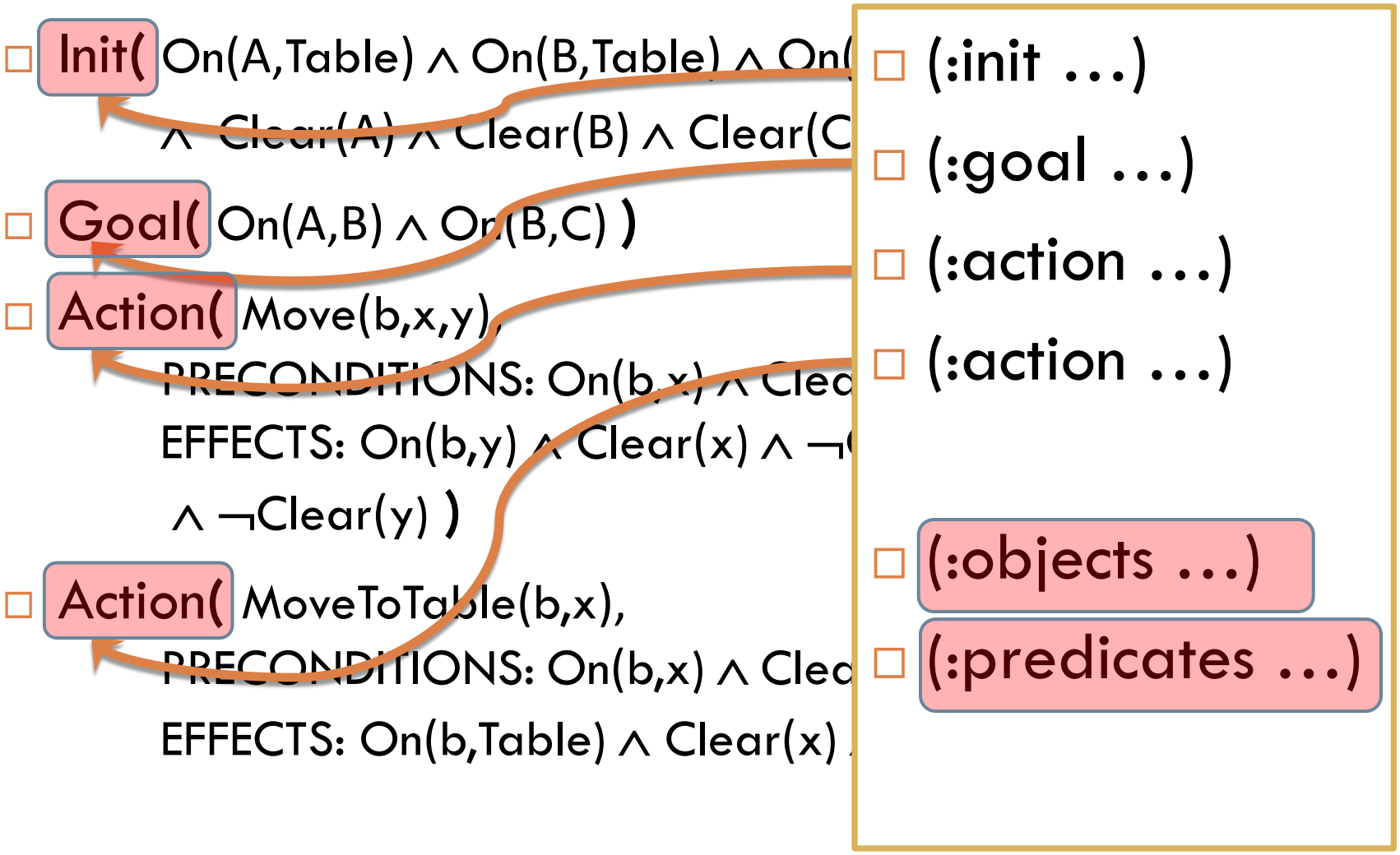
- Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table)
  ∧ Clear(A) ∧ Clear(B) ∧ Clear(C))
- Goal(On(A,B) ∧ On(B,C))
- Action(Move(b,x,y),
  PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
  EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x)
  ∧ ¬Clear(y))
- Action(MoveToTable(b,x),
  PRECONDITIONS: On(b,x) ∧ Clear(b)
  EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))

Planning Domain Description Language

- Init(On(A,Table) ∧ On(B,Table) ∧ On(
  ∧ Clear(A) ∧ Clear(B) ∧ Clear(C
- Goal(On(A,B) ∧ On(B,C))
- Action(Move(b,x,y),
  PRECONDITIONS: On(b,x) ∧ Clea
  EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬(
  ∧ ¬Clear(y))
- Action(MoveToTable(b,x),
  PRECONDITIONS: On(b,x) ∧ Clea
  EFFECTS: On(b,Table) ∧ Clear(x)

- (:init …)
- (:goal …)
- (:action …)
- (:action …)

Planning Domain Description Language



Planning Domain Description Language

- (:init …)
- (:goal …)
- (:action …)
- (:action …)

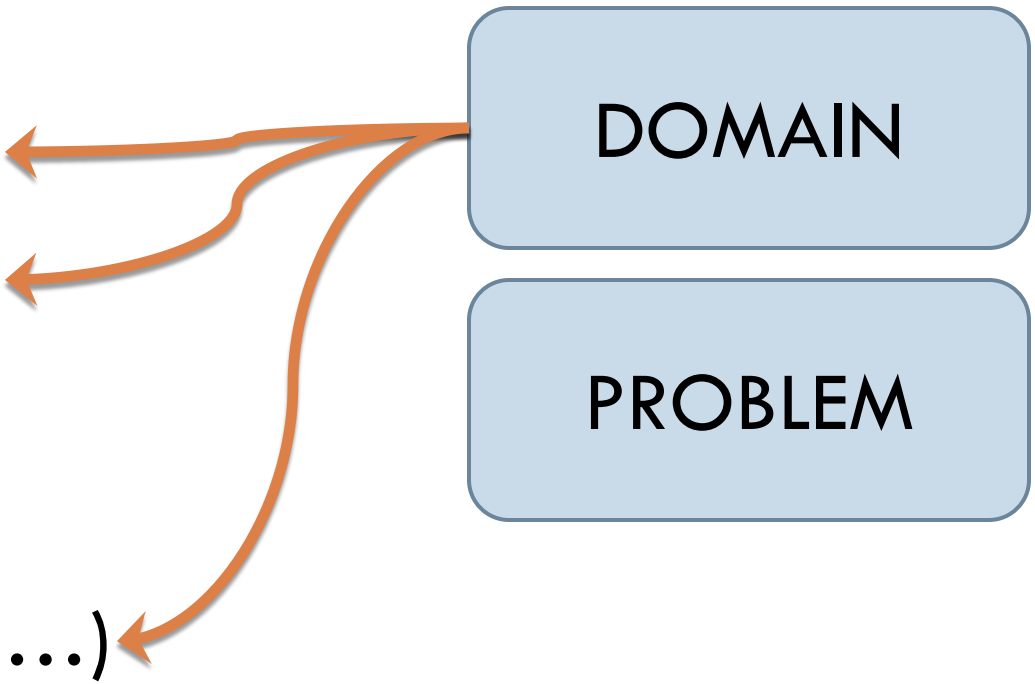
- (:objects …)
- (:predicates …)

DOMAIN

PROBLEM

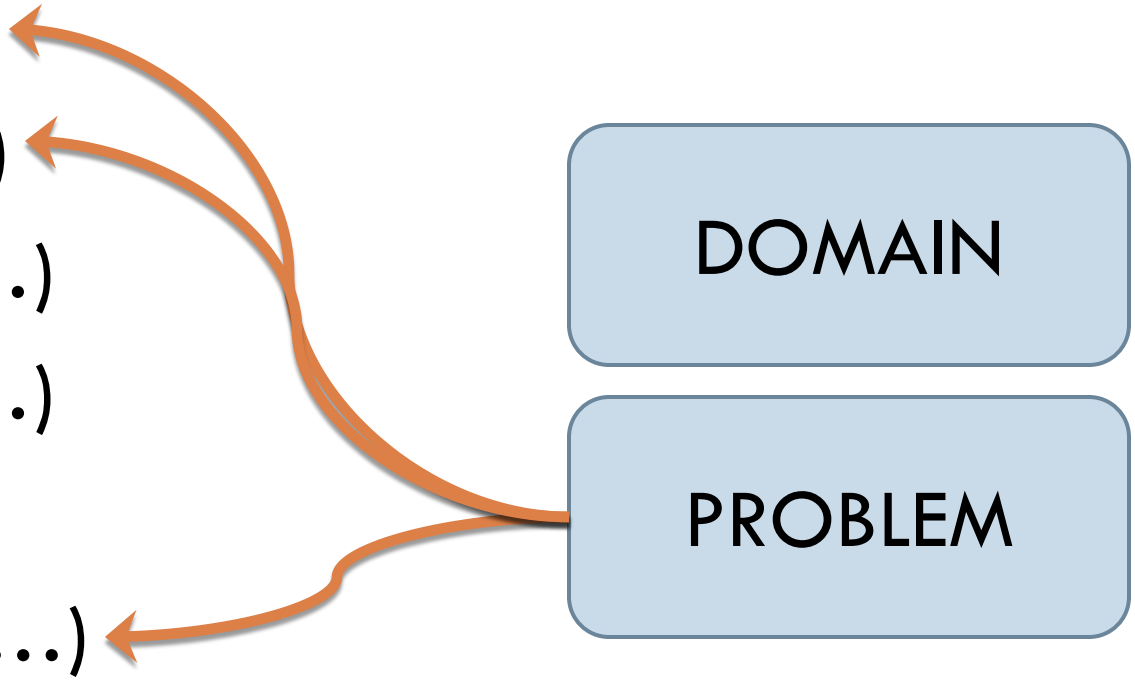
Planning Domain Description Language

- (:init …)
- (:goal …)
- (:action …)
- (:action …)
- (:objects …)
- (:predicates …)



Planning Domain Description Language

- (:init …)
- (:goal …)
- (:action …)
- (:action …)
- (:objects …)
- (:predicates …)



Planning Domain Description Language

- (:predicates …)
- (:action …)
- (:action …)

DOMAIN

- (:objects …)
- (:init …)
- (:goal …)

PROBLEM

Planning Domain Description Language

- On(A,B) → (on a b)
- ¬On(A,B) → (not (on a b))
- On(A,B) ∧ On(B,C) → (and (on a b) (on b c))
- On(x,y) → (on ?x ?y)

The blocks world example in PDDL

- Blocks world **domain**
  - Available predicates (:predicates …)
  - Available actions (:action …)

The blocks world example in PDDL

- Blocks world **domain**
  - Available predicates

```
(:predicates
        (on ?x ?y)  (clear ?x)
)
```

The blocks world example in PDDL

- Blocks world **domain**

  - Available action

```
(:action move
        :parameters (?b ?x ?y)
        :precondition (and
                (on ?b ?x) (clear ?b)
                (clear ?y))
        :effect (…)
)
```

The blocks world example in PDDL

- Blocks world **domain**

  - Available action

```
(:action move-to-table
        :parameters (?b ?x)
        :precondition (…)
        :effect (…)
)
```

The blocks world example in PDDL

- Blocks world **problem**
  - Available objects (:objects …)
  - Initial state (:init …)
  - Goal (:goal …)

The blocks world example in PDDL

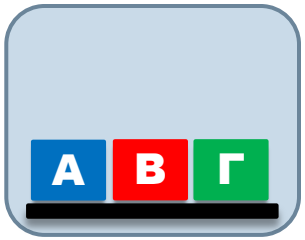
- Blocks world **domain**
  - Available objects

```
(:objects
        a b c table
)
```

The blocks world example in PDDL

- Blocks world **domain**

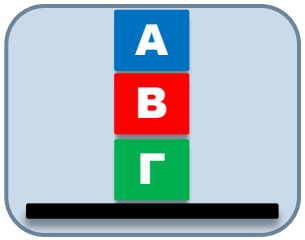
  - Initial state




```
(:init
        (on a table)  (clear a)
        (on b table)  (clear b)
        (on c table)   (clear c)
)
```

The blocks world example in PDDL

- Blocks world **domain**
  - Goal




```
(:goal
        (and (on a b) (on b c) )
)
```

The blocks world example in PDDL

blocks-domain.txt:

```
(define (domain gripper)
(:requirements :strips)
(:predicates (on ?x ?y)  (clear ?x))

(:action move
:parameters (?b ?x ?y)
:precondition (and (on ?b ?x) (clear ?b) (clear ?y))
:effect (and (not (on ?b ?x))
        (not (clear ?y))
        (on ?b ?y)
        (clear ?x)))
...
```

blocks-problem1.txt:

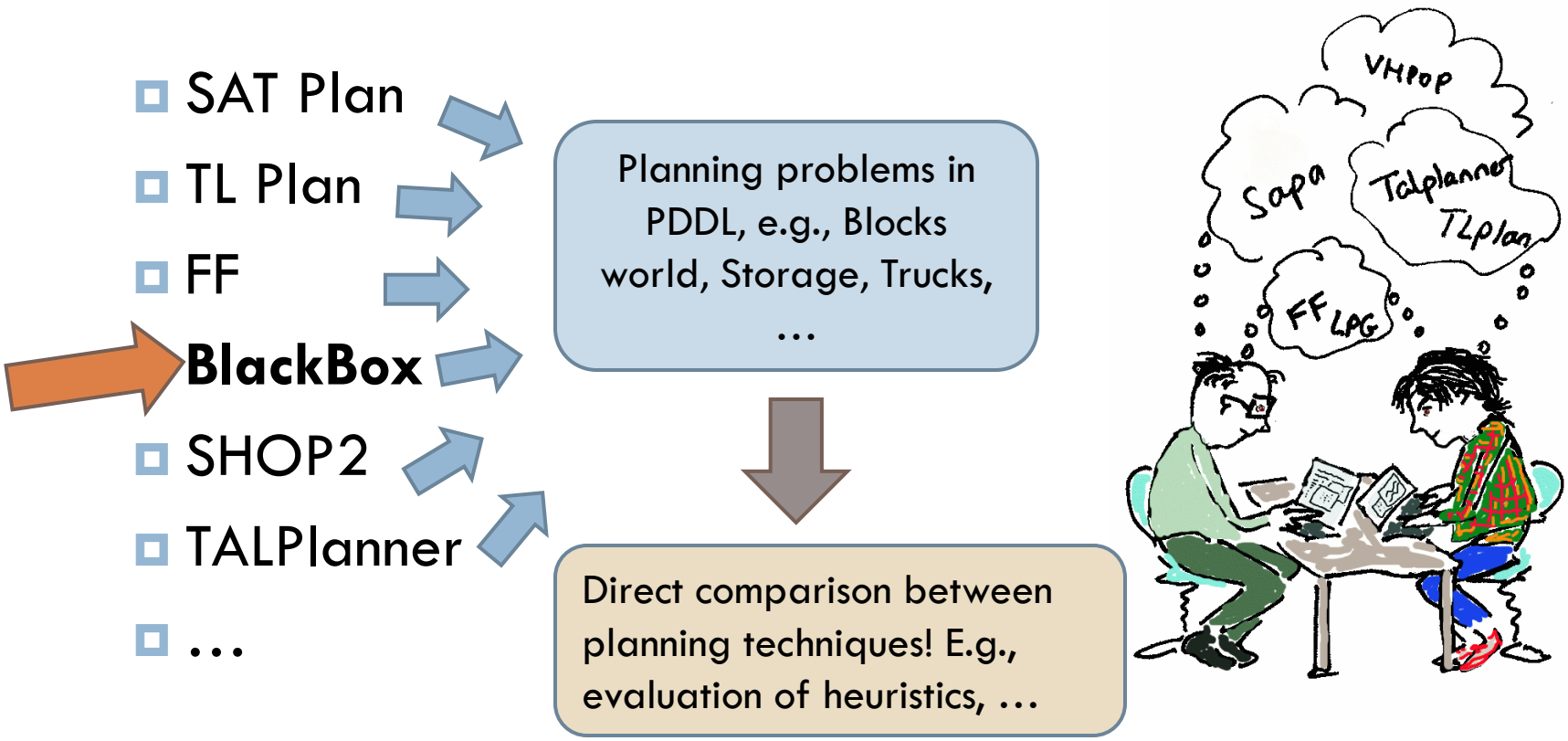
```
(define (problem gripper1)
(:domain gripper)
(:objects  a b c table)

(:init
  (on a table)   (on b table)  (on c table)
  (clear a)   (clear b)  (clear c)
)

(:goal (and (on a b) (on b c)))
)
```

Using PDDL planners

- Planning Domain Definition Language (PDDL)
  - International Planning Competition 1998 – today



Using PDDL planners: Blocks world

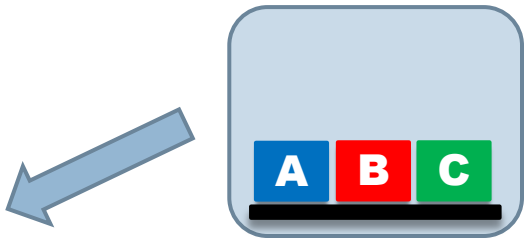
- blackbox –o blocks-domain.txt –f blocks-problem1.txt

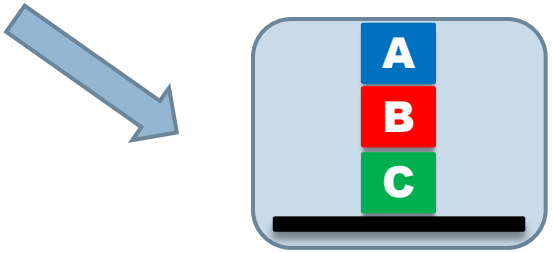
Begin plan

1 (move b table c)

2 (move a table b)

End plan





Using PDDL planners: Blocks world

- blackbox –o blocks-domain.txt –f blocks-problem2.txt

Begin plan

1 (move e b d)

2 (move b table e)

3 (move g f table)

4 (move f c g)

5 (move b e c)

6 (move e d f)

7 (move d a e)

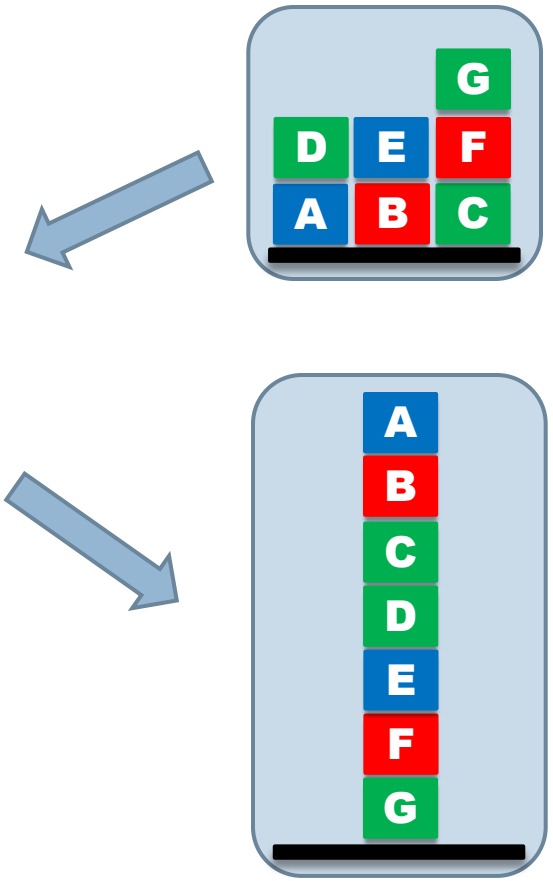
8 (move b c a)

9 (move c table d)

10 (move b a c)

11 (move a table b)

End plan

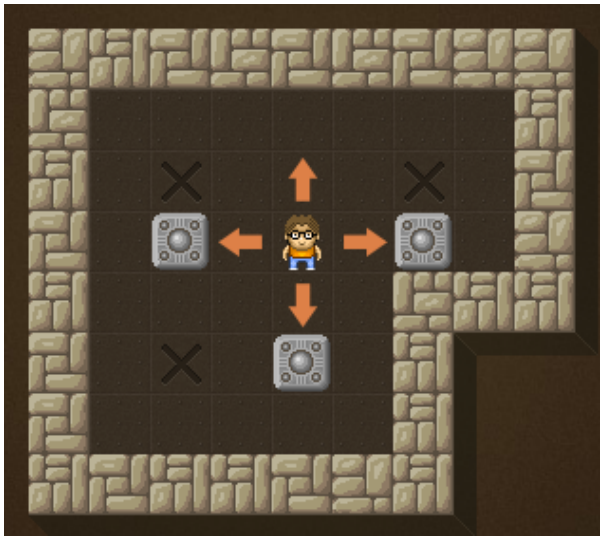


Using PDDL planners

- We can use blackbox (or any other off-the-self PDDL planner) for any problem we can write in PDDL!

Using PDDL planners: Sokoban

- We can use blackbox (or any other off-the-self PDDL planner) for any problem we can write in PDDL!


- Let's do this for Sokoban

Using PDDL planners: Sokoban

- Available predicates
- Available actions

- Available objects
- Initial state
- Goal

Using PDDL planners: Sokoban

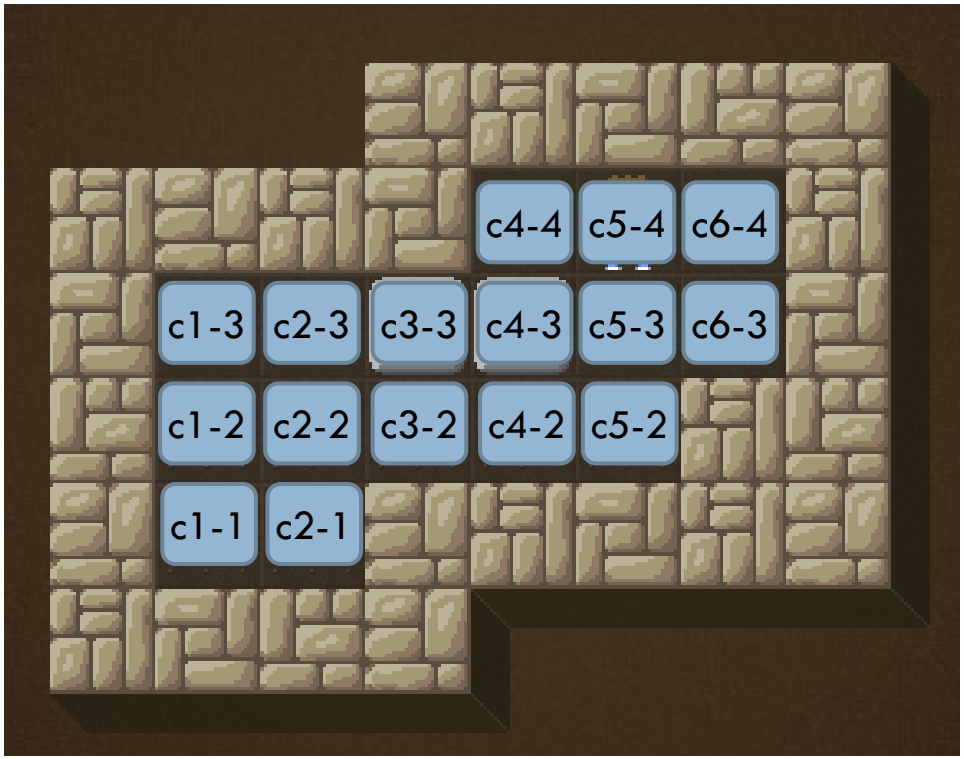
- Available predicates

Using PDDL planners: Sokoban

- Available predicates
  - (robot-at ?loc)
  - (object-at ?b ?loc)

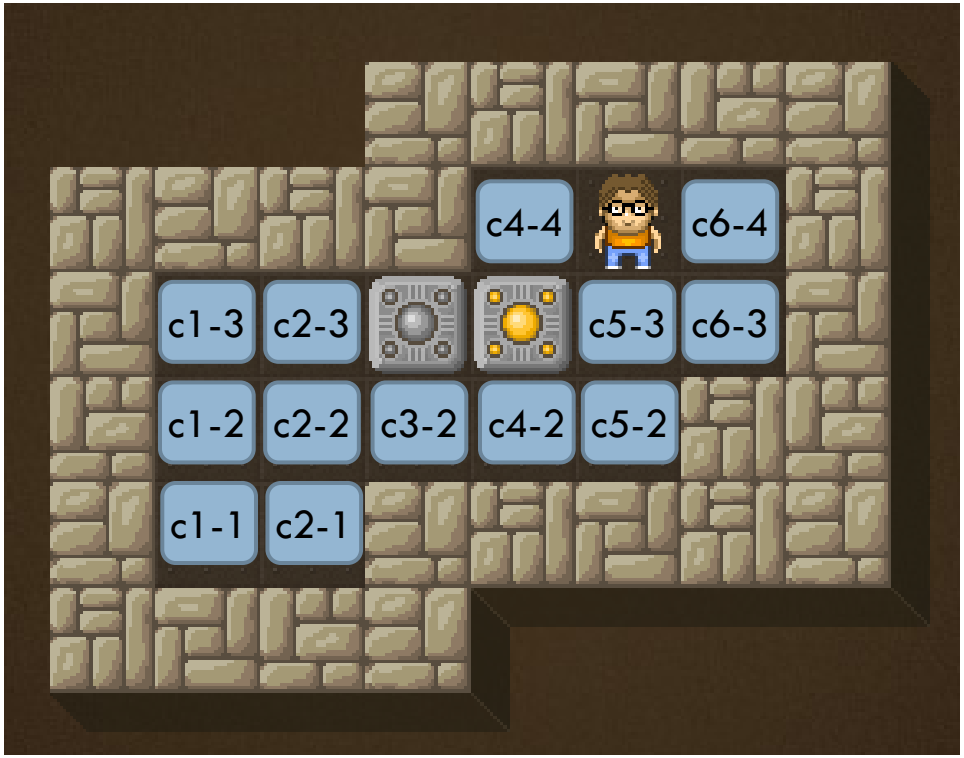
Using PDDL planners: Sokoban

- Available predicates
  - (robot-at ?loc)
  - (object-at ?b ?loc)
- Available objects
  - c1-1, c1-2, c2-1, …
  - box1, box2



Using PDDL planners: Sokoban

- Initial state
  - (robot-at c5-4)
  - (object-at box1 c3-3)
  - (object-at box2 c4-3)



Using PDDL planners: Sokoban

- Available actions
  - move(?from ?to ?dir)

  :precondition (

)

Using PDDL planners: Sokoban

- Available actions
  - move(?from ?to ?dir)

```
:precondition (and
   (robot-at ?from)
   (adjacent ?from ?to ?dir)
   (empty ?to)
)
```

Using PDDL planners: Sokoban

- Available actions
  - move(?from ?to ?dir)

:effect (and

)

Using PDDL planners: Sokoban

- Available actions
  - move(?from ?to ?dir)

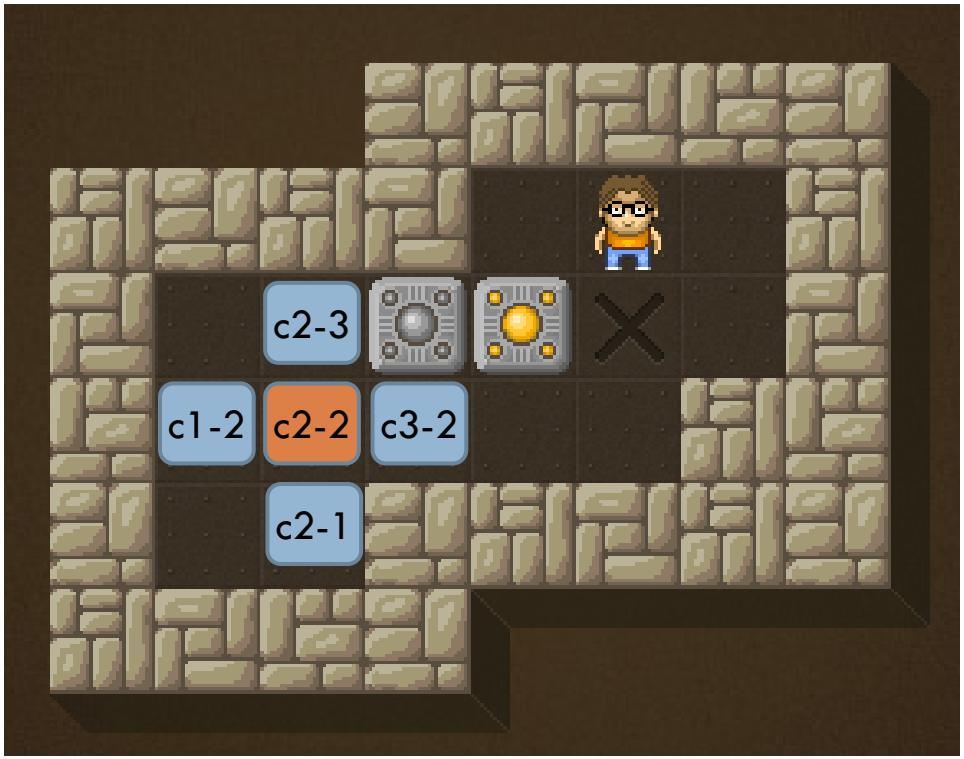
```
:effect (and
   (empty ?from)
   (robot-at ?to)
   (not (empty ?to))
   (not (robot-at ?from))
)
```

Using PDDL planners: Sokoban

- Available predicates
  - (robot-at ?loc)
  - (object-at ?b ?loc)
  - (adjacent ?from ?to ?dir)
  - (empty ?to)

Using PDDL planners: Sokoban

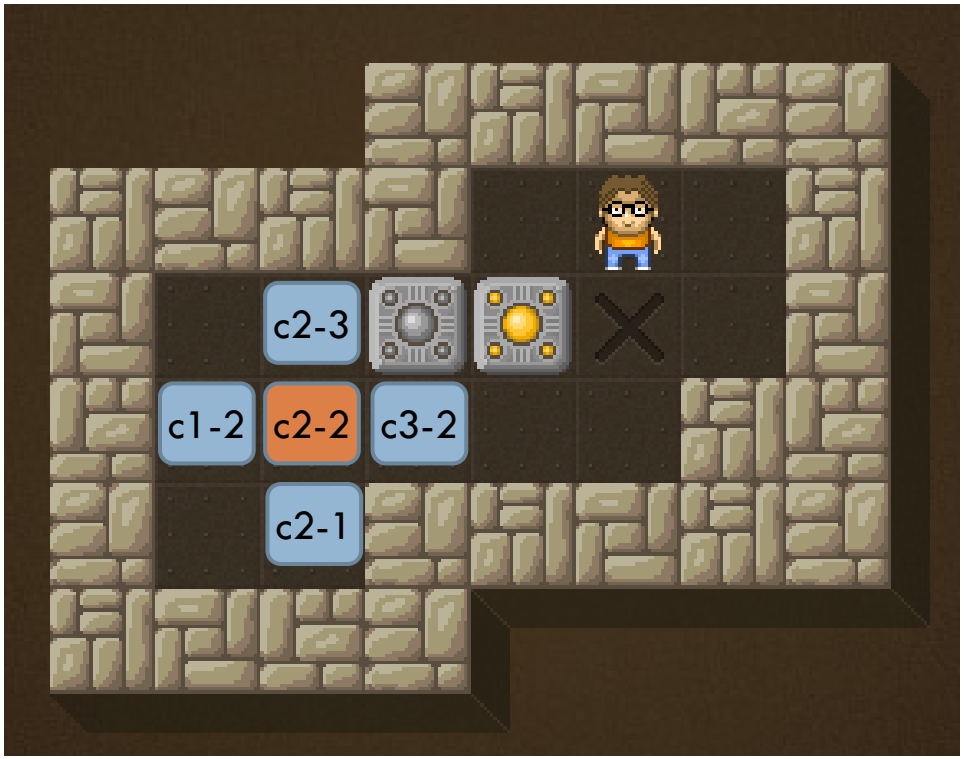
- Initial state
  - (robot-at c5-4)
  - (object-at box1 c3-3)
  - (object-at box2 c4-3)
  - (empty c1-1)
  - (empty c2-1)
  - (empty c1-2)
  - (empty c2-2)
  - …



Using PDDL planners: Sokoban

- Initial state
  - (robot-at c5-4)
  - (object-at box1 c3-3)
  - (object-at box2 c4-3)

  - (adjacent c2-2 c3-2 right)
  - (adjacent c2-2 c1-2 left)
  - (adjacent c2-2 c2-3 up)
  - (adjacent c2-2 c2-1 down)
  - …



Using PDDL planners: Sokoban

- Available actions
  - push(?rloc ?bloc ?floc ?dir ?b)

Using PDDL planners: Sokoban

- Available actions
  - push(?rloc ?bloc ?floc ?dir ?b)

```
:precondition (and
   (robot-at ?rloc)
   (object-at ?b ?bloc)
   (adjacent ?rloc ?bloc ?dir)
   (adjacent ?bloc ?floc ?dir)
   (empty ?floc)
)
```

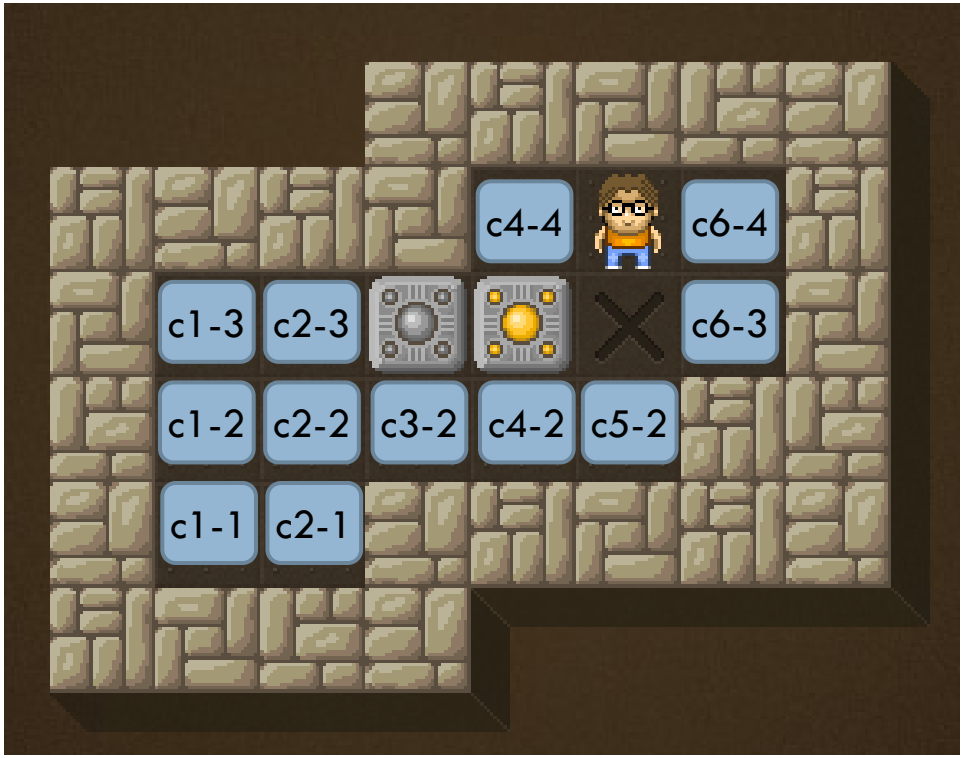
Using PDDL planners: Sokoban

- Available actions
  - push(?rloc ?bloc ?floc ?dir ?b)

```
:effect (and
   (robot-at ?bloc)
   (object-at ?b ?floc)
   (empty ?rloc)
   (not (robot-at ?rloc))
   (not (object-at ?b ?bloc))
   (not (empty ?floc))
)
```

Using PDDL planners: Sokoban

- Goal
  - (object-at box1 c4-3)
  - (object-at box2 c5-3)



Using PDDL planners: Sokoban

- blackbox –o sokoban-domain.txt –f sokoban-problem.txt

Begin plan

1 (push c4-4 c4-3 c4-2 down box1)

2 (push c4-3 c3-3 c2-3 left box2)

3 (move c3-3 c3-2 down)

4 (move c3-2 c2-2 left)

5 (move c2-2 c1-2 left)

…

27 (move c2-2 c1-2 left)

28 (move c1-2 c1-3 up)

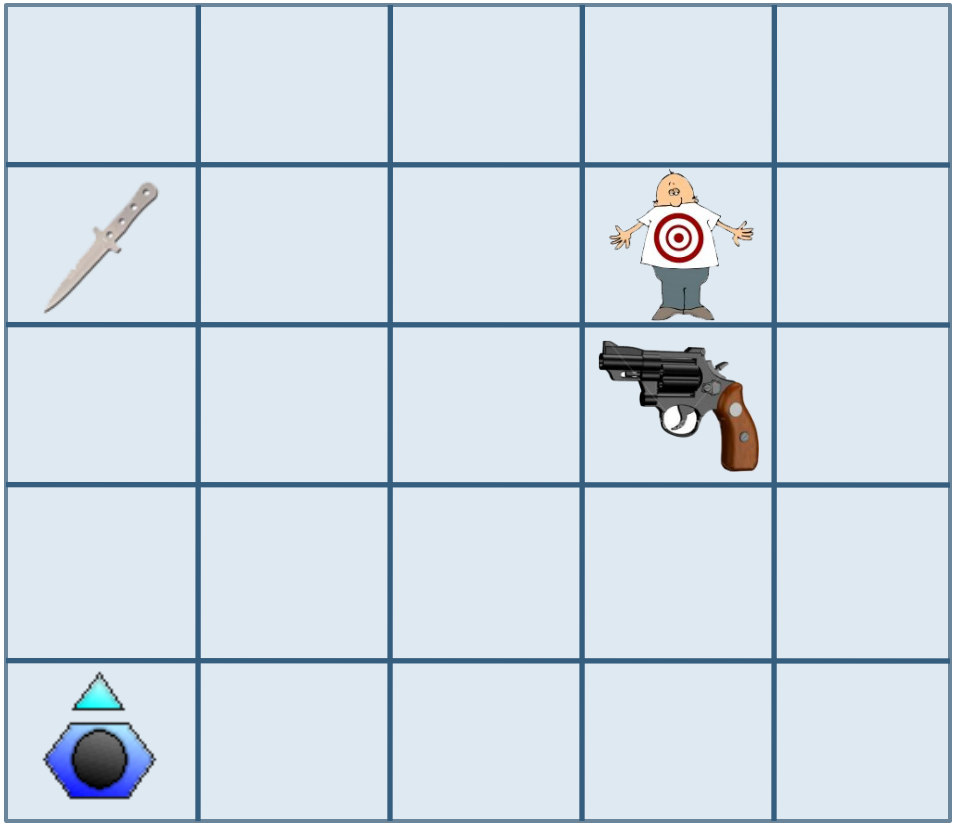
29 (push c1-3 c2-3 c3-3 right box1)

30 (push c2-3 c3-3 c4-3 right box1)

End plan

Using PDDL planners: SimpleGame

- SimpleGame domain



- turn(?fromd ?tod)
- move(?froml ?tol ?dir)
- pickup(?o ?l)
- stab(?l ?knife)
- shoot(?locn ?locp ?dir ?gun)

Building your own PDDL planner

- Pyperplan
  - Lightweight STRIPS planner written in Python. Developed during the planning course at Albert-Ludwigs-Universität Freiburg 2010/2011, GNU General Public License 3
  - https://bitbucket.org/malte/pyperplan
- Source code of academic planners
  - BlackBox: http://www.cs.rochester.edu/u/kautz/satplan/blackbox/
  - FastForward: http://www.loria.fr/~hoffmanj/ff.html
  - FastDownward: http://www.fast-downward.org/
- ANTLR Parser Generator
  - http://www.antlr.org/
  - http://www.antlr.org/grammar/1222962012944/Pddl.g

Building your own PDDL planner

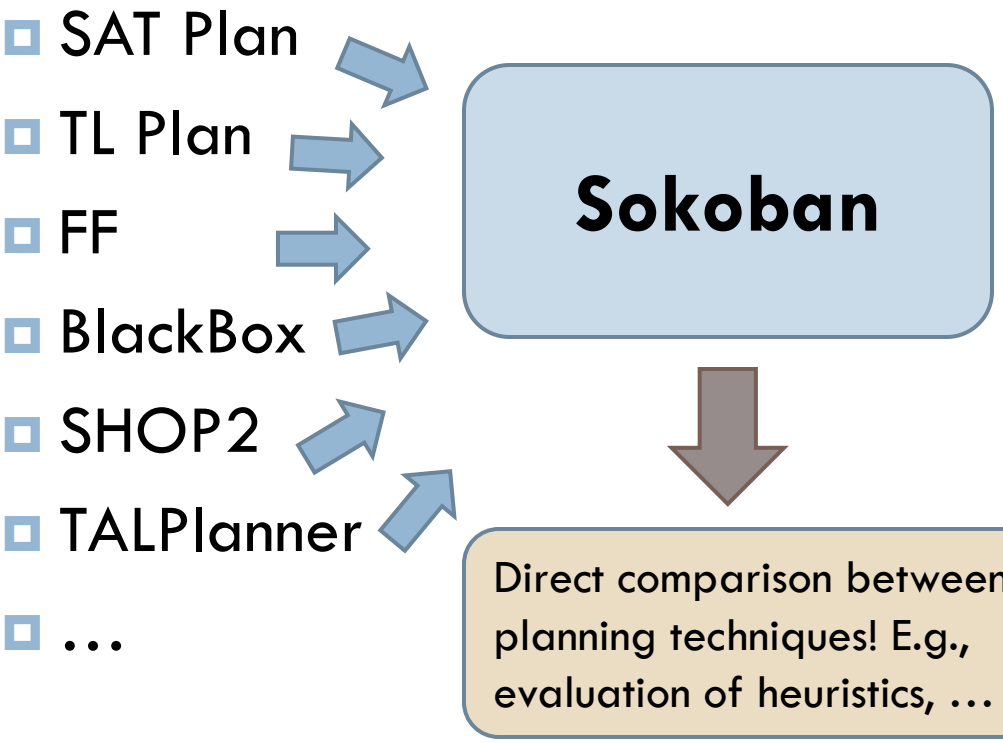
- Quick overview of the provided PROLOG code
- Good for quick prototyping ;-)

Building your own PDDL planner

- Quick overview of the provided PROLOG code
- Works with SWI Prolog 6.x (tested with 6.0.2)
- Predicates provided
  - parsePDDL(+DomainFile, +ProblemFile)
  - get_init(-InitialState)
  - get_goal(-Goal)
  - satisfies_goal(+State)
  - progress(+State, -Action, -NextState)
  - h(+State, -Value)
- {dfs,astar}planner(+DomainFile, +ProblemFile)

Planning Domain Description Language

- Planning Domain Definition Language (PDDL)
  - International Planning Competition 1998 – today



Using PDDL planners

- FastDownward [Helmert 2006]
  - Introduced a new type of heuristic that is not based on an empty list of negative actions
  - Efficient implementation of all notable forward search methods and heuristics (including the more recent ones)
  - Requires a number of preprocessing steps that transforms the STRIPS planning problem…

Using PDDL planners

- FastDownward [Helmert 2006]
  - Introduced a new type of heuristic that is not based on an empty list of negative actions
  - **Efficient implementation** of all notable **forward search** methods and **heuristics** (including the more recent ones)
  - Requires a number of preprocessing steps that transforms the STRIPS planning problem…
  - **Can be used as a framework to evaluate forward search planning methods**

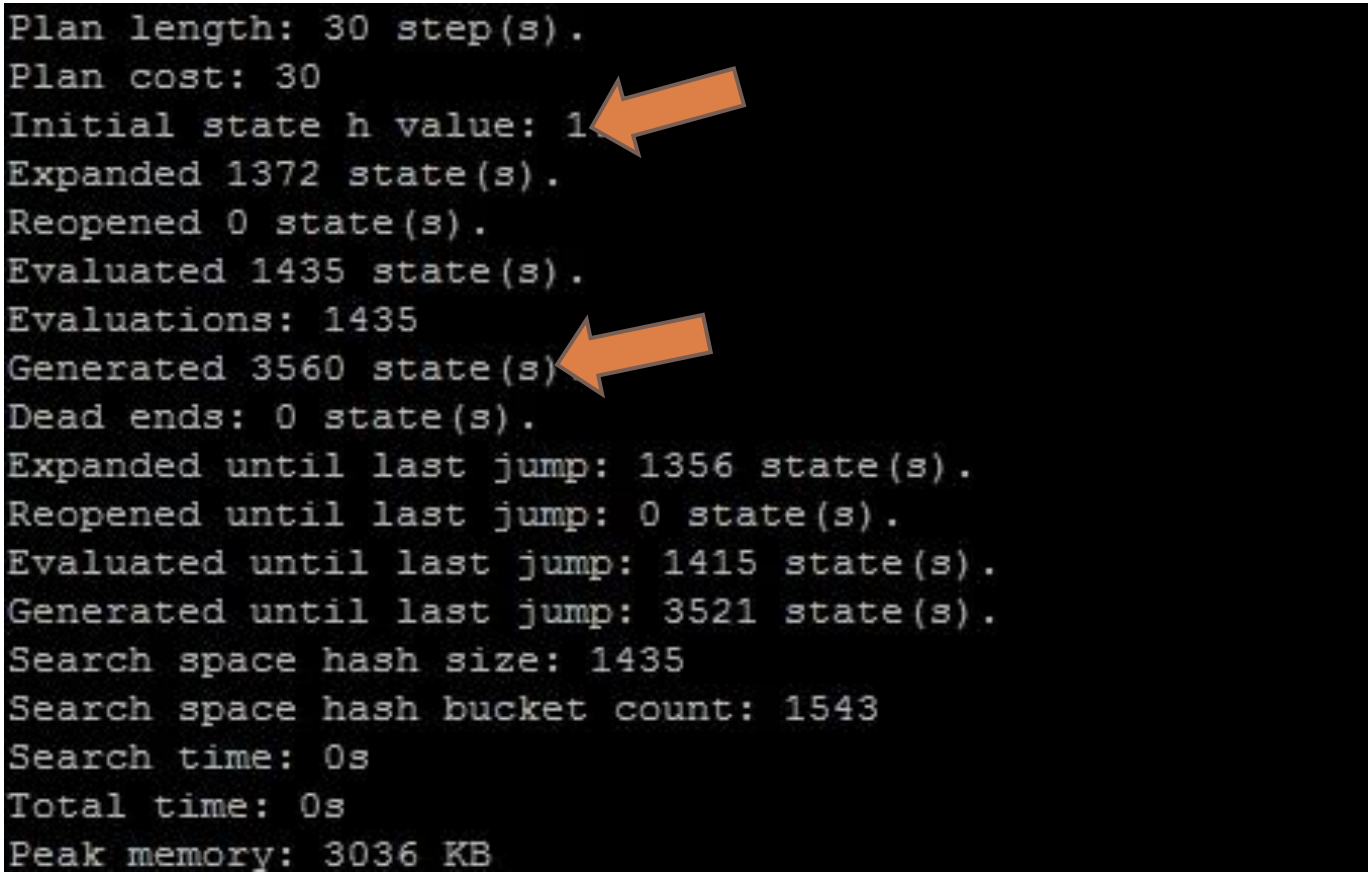
Using PDDL planners

- FastDownward [Helmert 2006]
  - translate/translate.py sokoban-domain.txt \
    sokoban.problem.txt
  - preprocess/preprocess < output.sas
  - search/downward --search SearchConfiguration < output

- http://www.fast-downward.org

Using PDDL planners: Sokoban

- search/downward --search "astar(blind())" <output



Using PDDL planners: Sokoban

- search/downward --search "astar(goalcount())"

```
Plan length: 30 step(s).
Plan cost: 30
Initial state h value: 1.
Expanded 1298 state(s).
Reopened 0 state(s).
Evaluated 1365 state(s).
Evaluations: 1365
Generated 3370 state(s).
Dead ends: 0 state(s).
Expanded until last jump: 1295 state(s).
Reopened until last jump: 0 state(s).
Evaluated until last jump: 1361 state(s).
Generated until last jump: 3365 state(s).
Search space hash size: 1365
Search space hash bucket count: 1543
Search time: 0s
Total time: 0s
Peak memory: 3040 KB
```

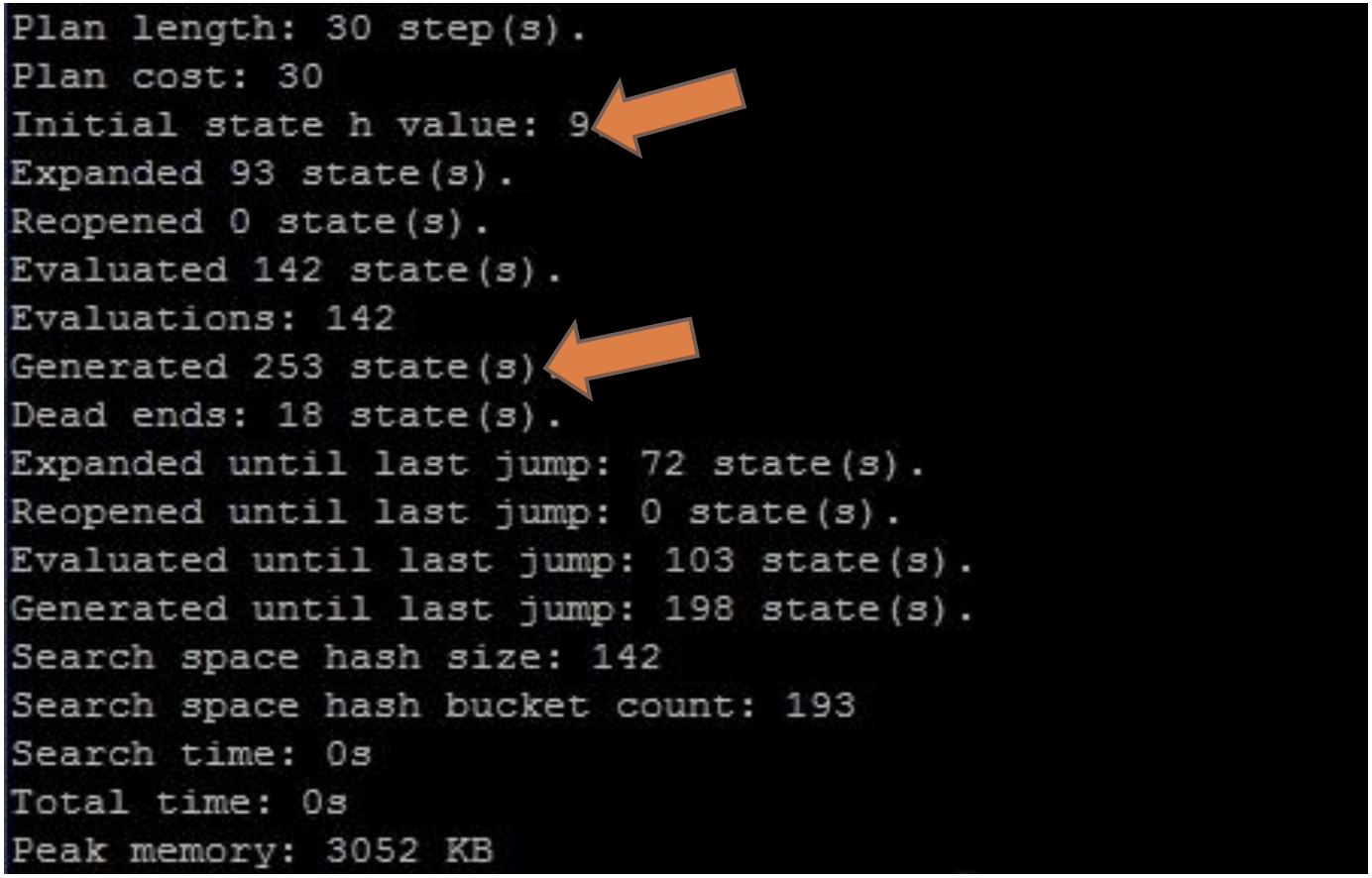
Using PDDL planners: Sokoban

- search/downward --search "astar(hmax())" <output

```
Plan length: 30 step(s).
Plan cost: 30
Initial state h value: 5
Expanded 139 state(s).
Reopened 0 state(s).
Evaluated 176 state(s).
Evaluations: 176
Generated 364 state(s).
Dead ends: 21 state(s).
Expanded until last jump: 133 state(s).
Reopened until last jump: 0 state(s).
Evaluated until last jump: 166 state(s).
Generated until last jump: 351 state(s).
Search space hash size: 176
Search space hash bucket count: 193
Search time: 0s
Total time: 0s
Peak memory: 3052 KB
```

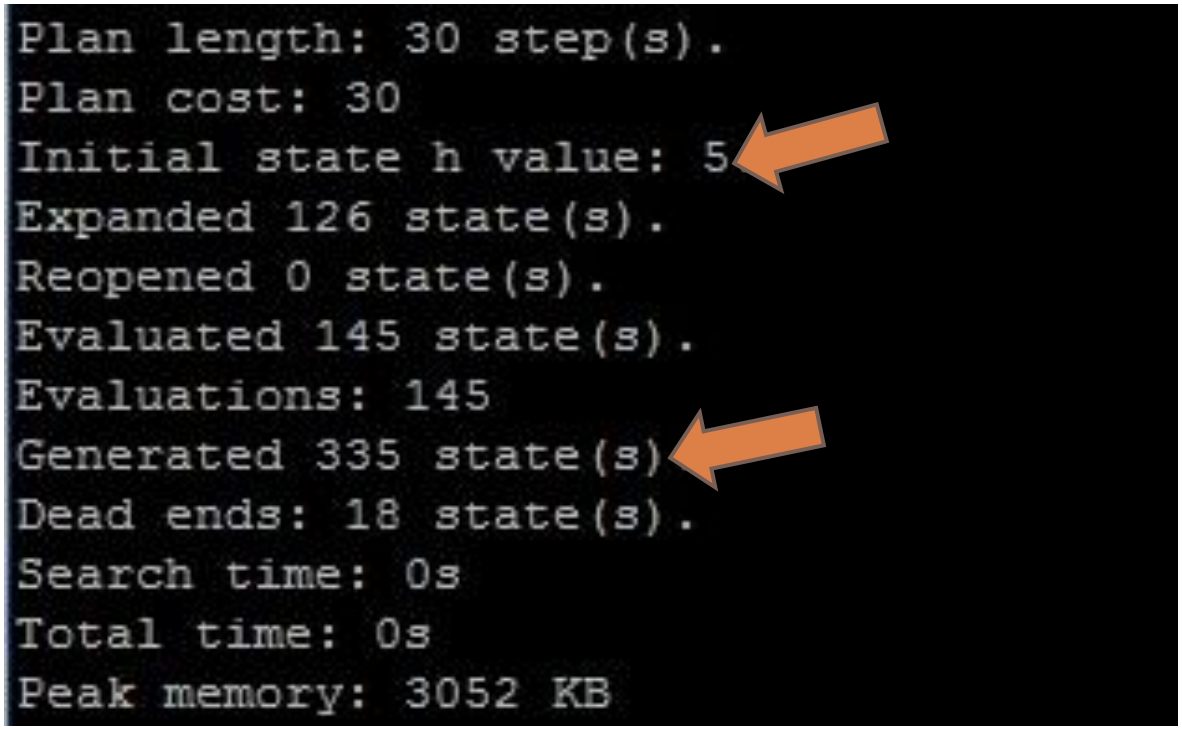
Using PDDL planners: Sokoban

- search/downward --search "astar(add())" <output



Using PDDL planners: Sokoban

- search/downward --search "lazy_greedy(ff())" <output



Using PDDL planners: Sokoban

- Notice that the planner does not know anything about the planning problem we are solving
- The heuristics we tried are **domain independent**
  - goal count
  - h_{max}
  - h_{add}
  - FF
- We will see how each one works in Lecture 4 (after we first go over planning graphs that are needed)

Next lecture

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

Bibliography

- Material
  - Artificial Intelligence: A Modern Approach 2nd Ed. Stuart Russell, Peter Norvig. Prentice Hall, 2003 Section 11.4

- References

  - PDDL - The Planning Domain Definition Language. Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins. Technical report, Yale Center for Computational Vision and Control, TR-98-003, 1998.
  - Unifying SAT-Based and Graph-Based Planning. Henry Kautz, Bart Selman. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1999
  - The Fast Downward Planning System. Malte Helmert. Artificial Intelligence Research (JAIR), Vol. 26, 2006