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: STRIPS planning, state-space search
- Lecture 2: Planning graphs, domain independent heuristics
- Lecture 3: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 4: Planning Domain Definition Language (PDDL), examples with planners and Prolog code
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

STRIPS planning

- What we have seen so far
  - The STRIPS formalism for specifying planning problems
  - Solving planning problems using state-based search
  - Progression planning
  - Effective heuristics for progression planning (based on relaxed problems, planning graphs)
  - PDDL tools for expressing and solving STRIPS problems

STRIPS planning

- What we have seen so far

Classical planning

  - There is **complete knowledge** about the initial state
  - Actions are **deterministic** with exactly one outcome
  - The solution is a **linear plan** (a sequence of actions)

STRIPS planning

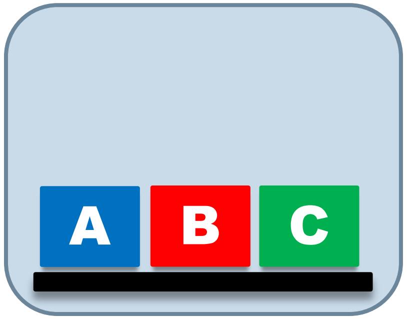
- What we have seen so far

Classical planning

  - There is **complete knowledge** about the initial state
  - Actions are **deterministic** with exactly one outcome
  - The solution is a **linear plan** (a sequence of actions)

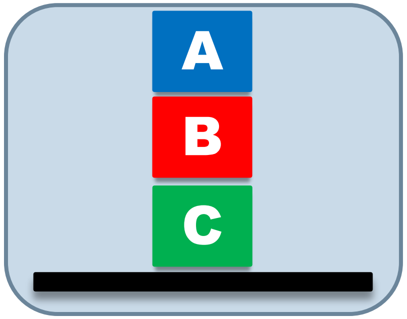
- Search “off-line”, then execute with “eyes closed”

STRIPS planning



On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

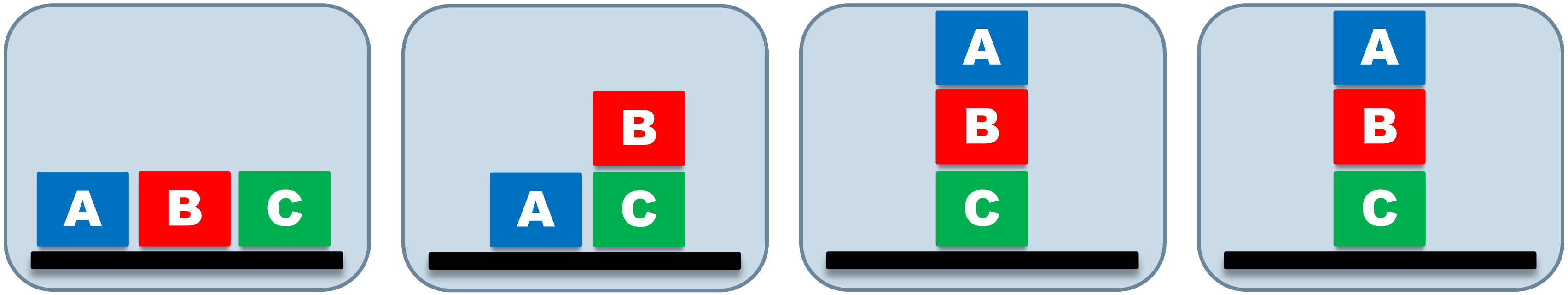
s_0



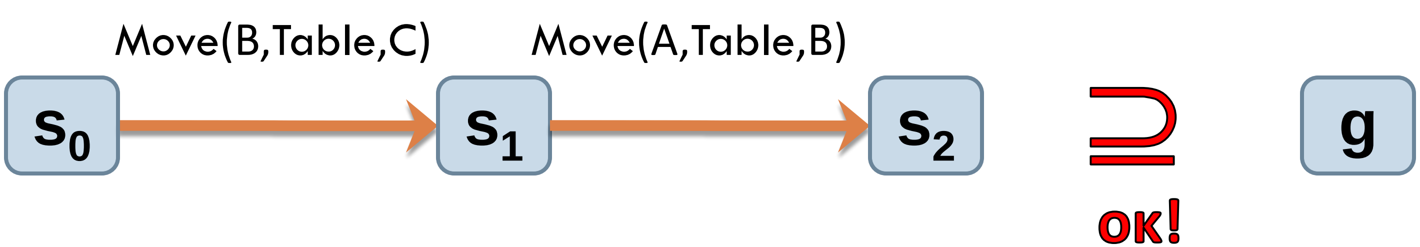
On(A,B)
On(B,C)

g

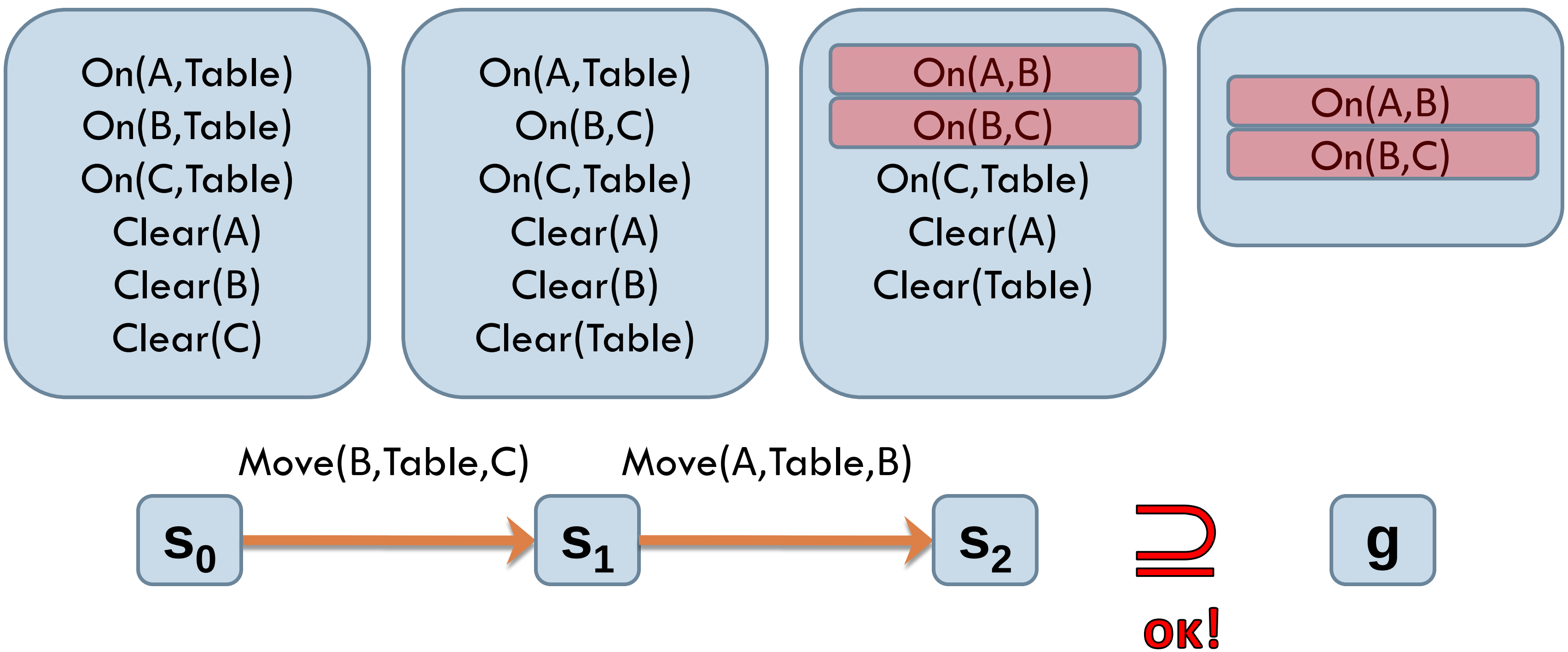
STRIPS planning



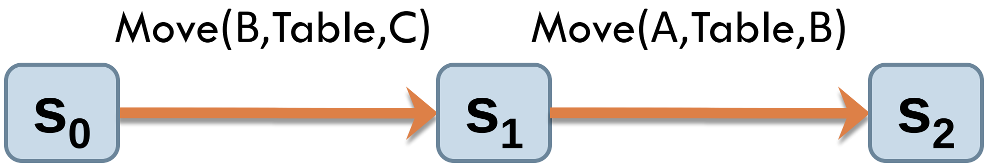
s_0	s_1	s_2	g
On(A,Table)	On(A,Table)	On(A,B)	On(A,B)
On(B,Table)	On(B,C)	On(B,C)	On(B,C)
On(C,Table)	On(C,Table)	On(C,Table)	
Clear(A)	Clear(A)	Clear(A)	
Clear(B)	Clear(B)	Clear(Table)	
Clear(C)	Clear(Table)		



STRIPS planning: Search



STRIPS planning: Execute



STRIPS planning: Execute

- 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

STRIPS planning: Execute

- 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

Planning beyond STRIPS

- What we have **not** seen so far

Planning beyond STRIPS

- What we have **not** seen so far
  - Initial state with **incomplete information**

Planning beyond STRIPS

- What we have **not** seen so far
  - Initial state with **incomplete information**
    - Open world assumption, e.g., I don't know anything about block D, could be sitting anywhere
    - Disjunctive information, e.g., $On(A,B) \vee On(B,A)$
    - Existential information, e.g., I know there is a block on top of A but I don't know which one: $\exists x\ On(x,A)$

Planning beyond STRIPS

- What we have **not** seen so far
  - Initial state with **incomplete information**
    - Open world assumption, e.g., I don't know anything about block D, could be sitting anywhere
    - Disjunctive information, e.g., $On(A,B) \vee On(B,A)$
    - Existential information, e.g., I know there is a block on top of A but I don't know which one: $\exists x\ On(x,A)$
    - Game-world: I know there is treasure hidden in some chest but I don't know which one

Planning beyond STRIPS

- What we have **not** seen so far
  - **Nondeterministic actions** with more than one outcome

Planning beyond STRIPS

- What we have **not** seen so far
  - **Nondeterministic actions** with more than one outcome
    - An action succeeds with a degree of probability, e.g., move(x,b,y) action succeeds with a 90% probability
    - An action may have more than one outcomes, e.g., moving a block may lead to moving the intended block or a neighbouring one

Planning beyond STRIPS

- What we have **not** seen so far
  - **Nondeterministic actions** with more than one outcome
    - An action succeeds with a degree of probability, e.g., move(x,b,y) action succeeds with a 90% probability
    - An action may have more than one outcomes, e.g., moving a block may lead to moving the intended block or a neighbouring one
    - Game-world: Picking a lock may result in the door opening or the tool breaking

Planning beyond STRIPS

- What we have **not** seen so far
  - Representation of the **duration** of actions

Planning beyond STRIPS

- What we have **not** seen so far
  - Representation of the **duration** of actions
    - How can we say that an action takes more time than another one?
    - How can we say that the goal should be reached within a time limit?

Planning beyond STRIPS

- What we have **not** seen so far
  - **Exogenous events**

Planning beyond STRIPS

- What we have **not** seen so far
  - **Exogenous events**
    - What if in the blocks world we decided to push one of the blocks from time to time and change its location?
    - What if in the blocks world there was another gripper that could move blocks in order to achieve their goal?

Planning beyond STRIPS

- What we have **not** seen so far
  - **Exogenous events**
    - What if in the blocks world we decided to push one of the blocks from time to time and change its location?
    - What if in the blocks world there was another gripper that could move blocks in order to achieve their goal?
    - Game-world: the state of the game is altered not only by the moves of our agent/NPC but also by the human player and other agents

Planning beyond STRIPS

- What we have **not** seen so far
  - **Sensing actions**

Planning beyond STRIPS

- What we have **not** seen so far
  - **Sensing actions**
    - These actions do not affect the world but instead the knowledge of the agent about the world is updated
    - E.g., sense which is the block that is on top of block A

Planning beyond STRIPS

- What we have **not** seen so far
  - **Sensing actions**
    - These actions do not affect the world but instead the knowledge of the agent about the world is updated
    - E.g., sense which is the block that is on top of block A
    - Game-world: look-inside(chest1) could update the information that the agent has about what is lying inside the chest

Planning beyond STRIPS

- What we have **not** seen so far
  - A **more expressive solution**
    - Looking for a linear plan is the simplest case (and works well only in classical planning problems)

Planning beyond STRIPS

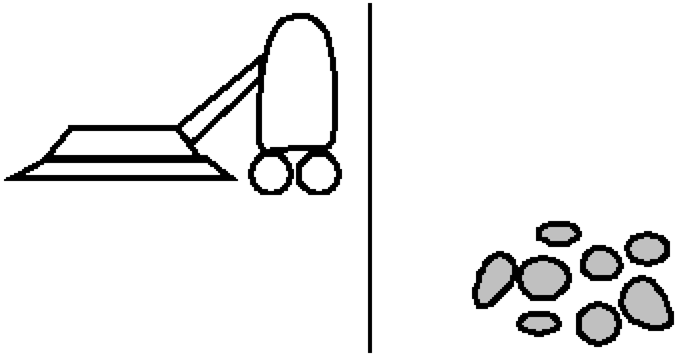
- What we have **not** seen so far
  - A **more expressive solution**
    - Looking for a linear plan is the simplest case (and works well only in classical planning problems)
  - A solution can be
    - a tree of nested if-then-else statements, e.g., [if open(chest) then … else …]
    - a more expressive program that specifies how the agent should behave, e.g., [while ¬open(chest) do … end while]

Planning beyond STRIPS

- Let's see some scenarios that combine such features

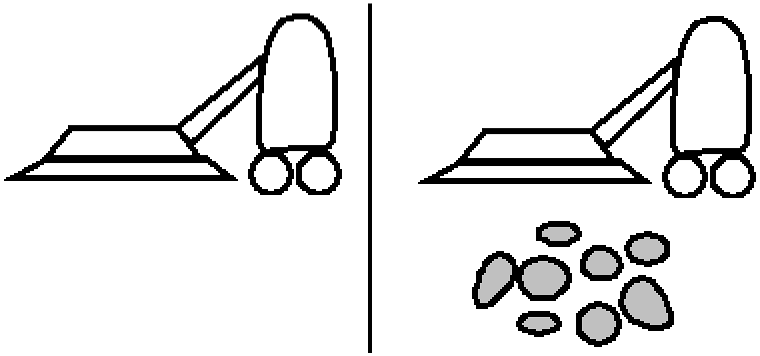
Planning beyond STRIPS

- Three versions of the Vacuum Cleaner domain



Planning beyond STRIPS

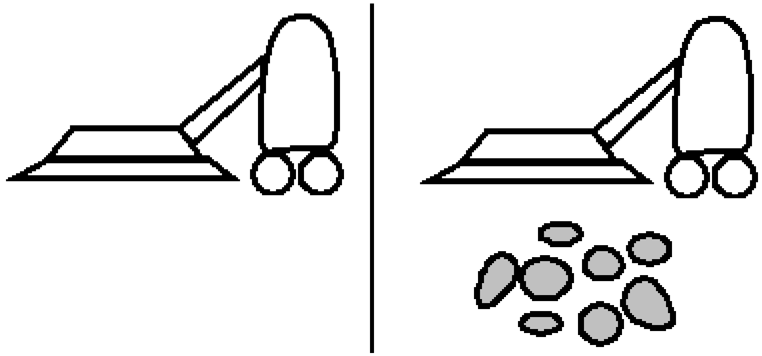
- **Version 1** of the Vacuum Cleaner domain



  - Incomplete information about the initial state
    - The cleaning bot does not know its position
  - Deterministic actions
    - Actions moveLeft, moveRight, clean always succeed with the intuitive effects
  - The bot does not get any other information about the state

Planning beyond STRIPS

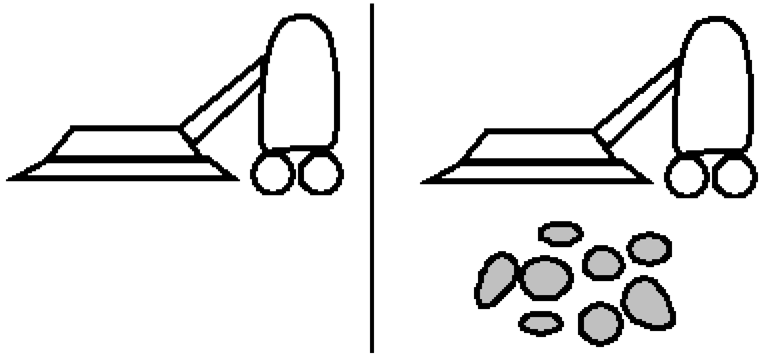
- **Version 1** of the Vacuum Cleaner domain



- Conformant planning
  - Find a **sequence of actions** that achieves the goal in **all possible cases**

Planning beyond STRIPS

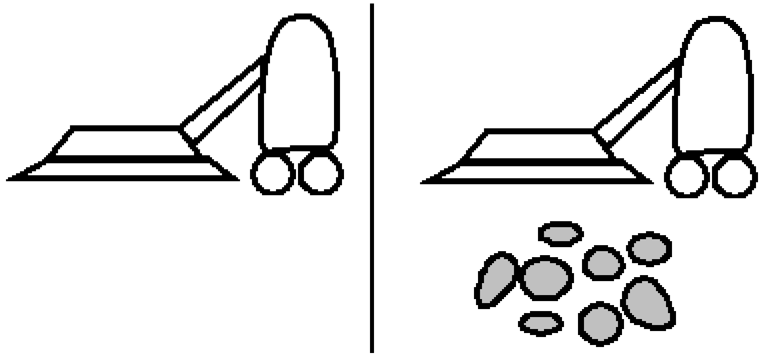
- **Version 1** of the Vacuum Cleaner domain



- Conformant planning
  - Find a **sequence of actions** that achieves the goal in **all possible cases**
  - **Plan:** [moveLeft, clean, moveRight, clean]

Planning beyond STRIPS

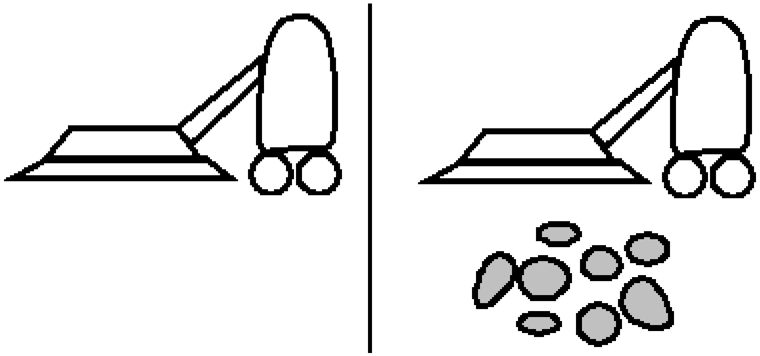
- **Version 2** of the Vacuum Cleaner domain



  - Incomplete information about the initial state
    - The cleaning bot does not know its position
  - Deterministic actions
    - Actions moveLeft, moveRight, clean always succeed with the intuitive effects
  - At run-time the cleaning bot can see which state it is in

Planning beyond STRIPS

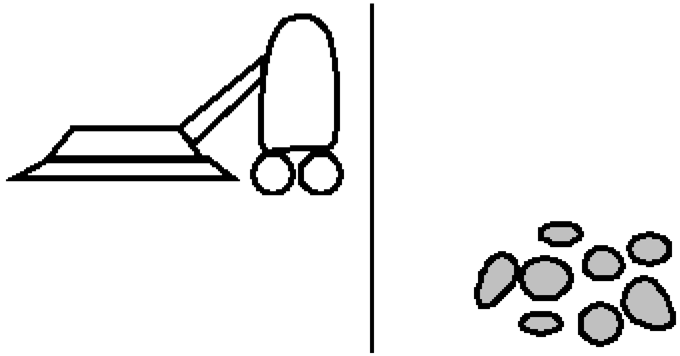
- **Version 2** of the Vacuum Cleaner domain



- Conditional planning
  - Find a plan that also uses **if-then-else** statements, such that it achieves the goal assuming that conditions can be evaluated at run-time
  - **Plan:** [**if** isRight **then** clean **else** moveRight, clean]

Planning beyond STRIPS

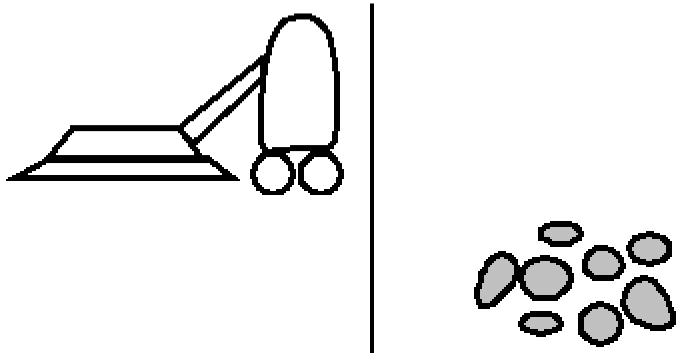
- **Version 3** of the Vacuum Cleaner domain



  - Complete information about the initial state
    - The cleaning bot is on the left, there is dirt on the right
  - Nondeterministic actions
    - Actions moveLeft, moveRight my fail without affecting the state
  - At run-time the cleaning bot can see which state it is in

Planning beyond STRIPS

- **Version 3** of the Vacuum Cleaner domain



- Planning for more expressive plans
  - Find a a plan that also uses **while** statements, such that it eventually achieves the goal assuming that conditions can be evaluated at run-time
  - **Plan:** [**while** isLeft **do** moveRight **end while**, clean]

Planning beyond STRIPS

- We see that the resulting plan need not be a linear sequence of actions

- How do we search for such plans?
  - AIMA Section 12.3: Planning and acting in nondeterministic domains
  - AIMA Section 12.4: Conditional planning

Planning beyond STRIPS

- We see that the resulting plan need not be a linear sequence of actions

- How do we search for such plans?
  - AIMA Section 12.3: Planning and acting in nondeterministic domains
  - AIMA Section 12.4: Conditional planning

  - Let's see an interesting extension of STRIPS that aims to account for some of the problems we identified

Planning beyond STRIPS

- Planning with Knowledge and Sensing (PKS)
  - [Petrick, Bacchus 2002]
  - http://homepages.inf.ed.ac.uk/rpetrick/software/pks/
- Extension of STRIPS that takes into account that some information will be available at run-time
  - K_f is like the normal STRIPS database but with open world
  - K_w holds literals whose truth value will be known at run-time
  - K_v holds literals with terms that will be known at run-time
  - K_x holds exclusive or information about literals
- Works with conditional plans that take cases

Planning beyond STRIPS

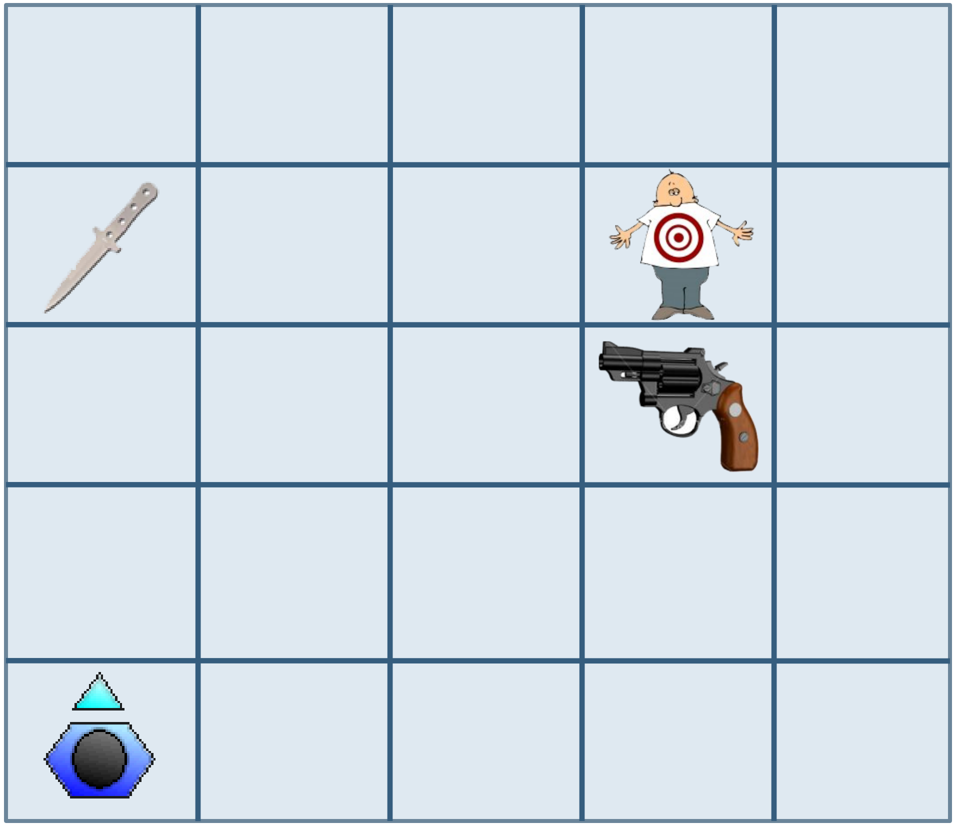
- We see that the resulting plan need not be a linear sequence of actions
- How do we search for such plans?
  - AIMA Section 12.3: Planning and acting in nondeterministic domains
  - AIMA Section 12.4: Conditional planning
- **Are these enough for building a real NPC?**

Planning beyond STRIPS

- What happens when an **exogenous event** changes something in the state while a plan is executed?

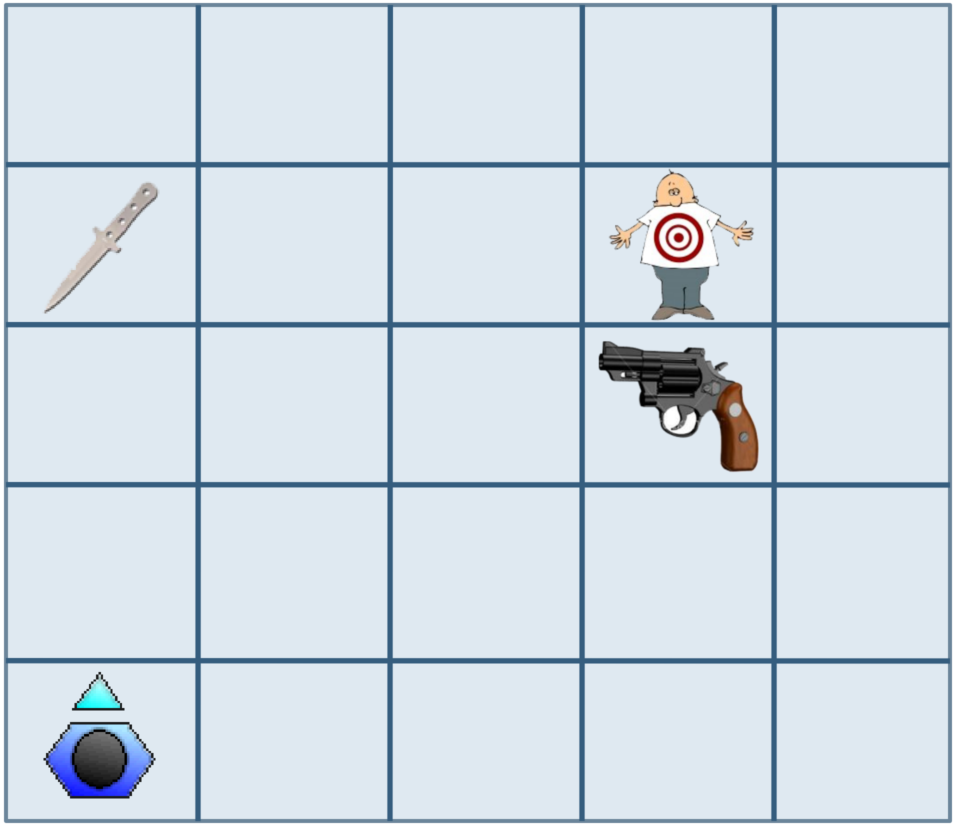
Planning beyond STRIPS

- MiniGame domain



Planning beyond STRIPS

- MiniGame domain



- up
- up
- up
- pickup
- right
- right
- right
- stab

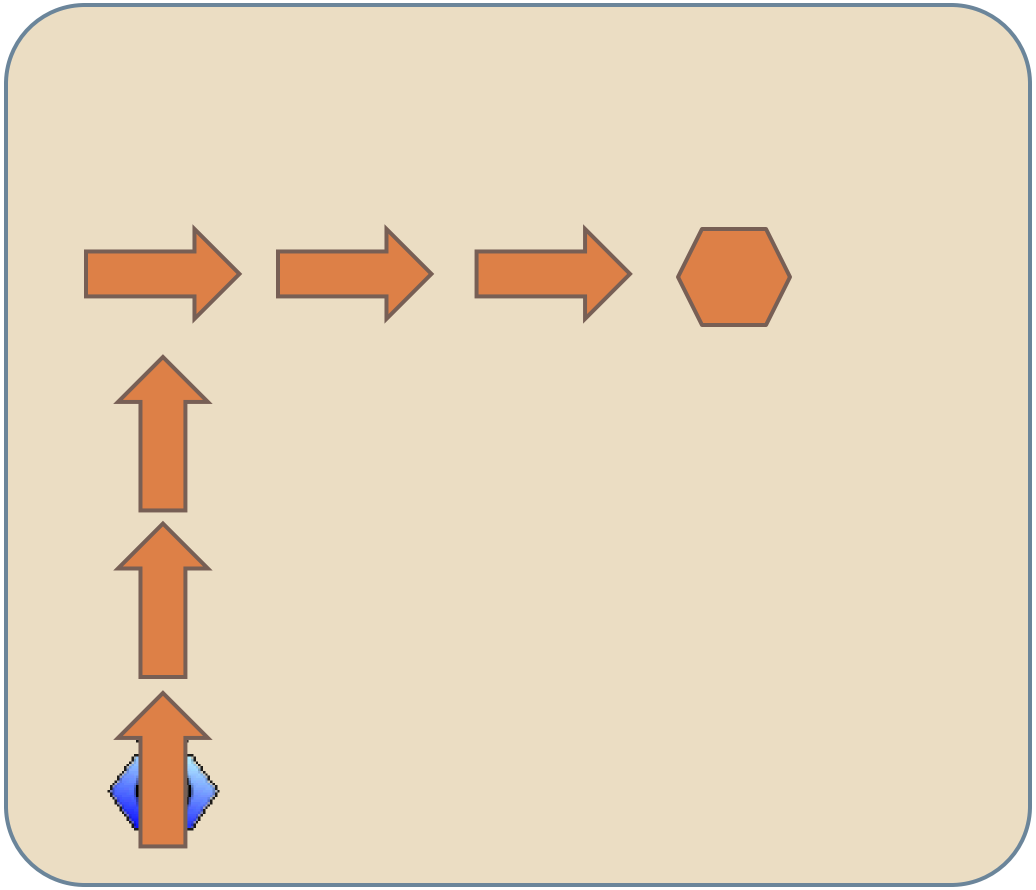
Planning beyond STRIPS

- MiniGame domain

- up
- up
- up
- pickup
- right
- right
- right
- stab

Planning beyond STRIPS

- MiniGame domain



- up
- up
- up
- pickup
- right
- right
- right
- stab

Planning beyond STRIPS

- MiniGame domain

Planning beyond STRIPS

- What happens when an **exogenous event** changes something in the state while a plan is executed?
  - The human player picks up the weapon that was part of the plan for the NPC
  - The player pushes the NPC out of the position it thinks its located
  - ...

Planning beyond STRIPS

- What happens when an **exogenous event** changes something in the state while a plan is executed?
  - Before executing the next action check that the preconditions of the actions are satisfied
  - Before executing the next action check that the preconditions of all remaining actions will be satisfied
  - Specify some special conditions that should hold at each step of the plan in order to continue executing it

Planning beyond STRIPS

- What happens when an **exogenous event** changes something in the state while a plan is executed?
  - Before executing the next action check that the preconditions of the actions are satisfied
  - Before executing the next action check that the preconditions of all remaining actions will be satisfied
  - Specify some special conditions that should hold at each step of the plan in order to continue executing it

- AIMA Section 12.5: Execution monitoring and replanning

Planning beyond STRIPS

- The approaches we have seen so far look for a plan that features simple programming constructs

Planning beyond STRIPS

- The approaches we have seen so far look for a plan that features simple programming constructs
- What if we could also provide the planner with a "**sketch**" of **how the plan should look like**?
  - Note that this makes sense only for a particular application, i.e., it is domain dependant

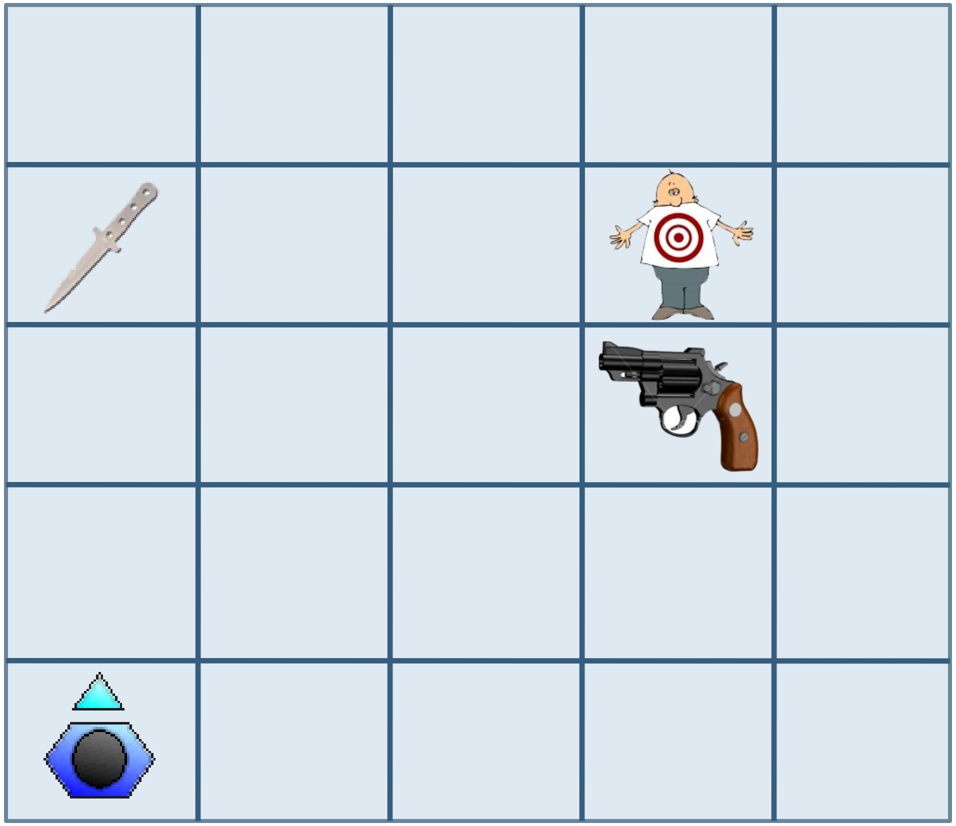
Planning beyond STRIPS

- The approaches we have seen so far look for a plan that features simple programming constructs
- What if we could also provide the planner with a "**sketch**" of **how the plan should look like**?
  - Note that this makes sense only for a particular application, i.e., it is domain dependant

- In this way we can also specify a behavior for an agent that works in an **"on-line"** manner
  - First, **find** a way to get a weapon. **Execute** the plan.
  - **Then, find** a way to get to the player. …

Planning beyond STRIPS

- MiniGame domain



Planning beyond STRIPS

- Golog: High-level agent programming language
  - search (

    (turn; π x. move(x))*;

    π x. pick-up(x);

    ?(π x. gun(x) and npc-holding(x));

    );

    search (

    (turn; π x. move(x))*;

    ?(npc-at(x) and player-at(y) and adjacent (x,y));

    );

    shoot-player

Planning beyond STRIPS

- Golog: High-level agent programming language

$\alpha,$	primitive action
$\phi?,$	wait or test for a condition
$\delta_1 ; \delta_2,$	sequence
$\delta_1 \mid \delta_2,$	nondeterministic branch
$\pi\, x.\, \delta(x),$	nondeterministic choice of argument
$\delta^*,$	nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endIf**,	conditional
while ϕ **do** δ **endWhile**,	while loop
$\delta_1 \parallel \delta_2,$	concurrency with equal priority
$\delta_1 \rangle\rangle\, \delta_2,$	concurrency with δ_1 at a higher priority
$\delta^{\parallel},$	concurrent iteration
$\langle\, \vec{x} : \phi(\vec{x}) \longrightarrow \delta(\vec{x})\, \rangle,$	interrupt
$p(\vec{\theta}).$	procedure call

Planning beyond STRIPS

- Golog: High-level agent programming language
  - Based on situation calculus, a first-order logic formalism
  - Much more expressive than STRIPS for specifying a domain and an initial situation
  - Many extensions in the literature, and a few working systems available, e.g.,
    - http://www.cs.toronto.edu/cogrobo/main/systems/index.html

Course overview

- Lecture 1: STRIPS planning, state-space search
- Lecture 2: Planning graphs, domain independent heuristics
- Lecture 3: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 4: Planning Domain Definition Language (PDDL), examples with planners and Prolog code
- 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 Sections 11.2, 12.3, 12.4, 12.5
- References

  - A knowledge-based approach to planning with incomplete information and sensing. Ronald P. A. Petrick, Fahiem Bacchus. In Proceedings of the International Conference on AI Planning and Scheduling Systems (AIPS), 2002
  - Golog: A Logic Programming Language for Dynamic Domains. Hector J. Levesque, Raymond Reiter, Yves Lesperance, Fangzhen Lin, Richard B. Scherl. Logic Programming, Vol. 31, No. 1-3. 1997