CS440/ECE448: Intro to Artificial Intelligence

Lecture 11: Planning

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

What is planning?

What is planning?

Plan = ‘*plan of attack*’:
Use inference to find a sequence of actions to reach a goal state from the initial state

Combines logic and search:

– Logic: to describe states and define actions
– Search: to find the actual sequence of actions

Applications of planning

– Space exploration
– Manufacturing
– Games (Bridge)
– Scheduling
– Logistics
– Semantic web support

Main types of planners

Domain-specific:

Tuned to target domain; don't generalize;
used in real-world applications

(*In CS440*): **domain-independent planning**

the only domain-specific knowledge:
definitions of basic actions;
requires many simplifying assumptions;
= *classical planning*

Classical planning: assumptions

The environment is:

– Fully observable
– Deterministic
– Static
– Known
– Finite (finitely many states and actions)

Under these assumptions,
a plan is a linear sequence of actions,
and planning can be done off-line

Classical planning

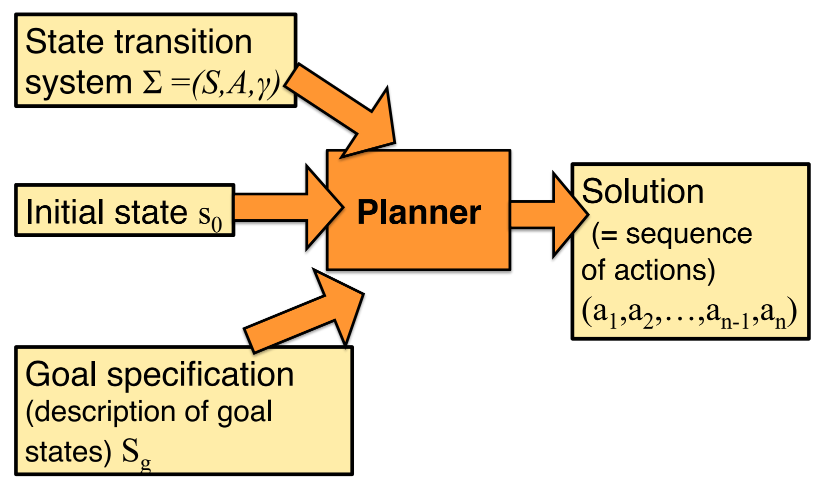
State transition system $\Sigma = (S, A, \gamma)$

– $S = \{states\}$
– $A = \{actions\}$
– $\gamma = S \times A \rightarrow 2^S$ $\{state\ transition\ function)$

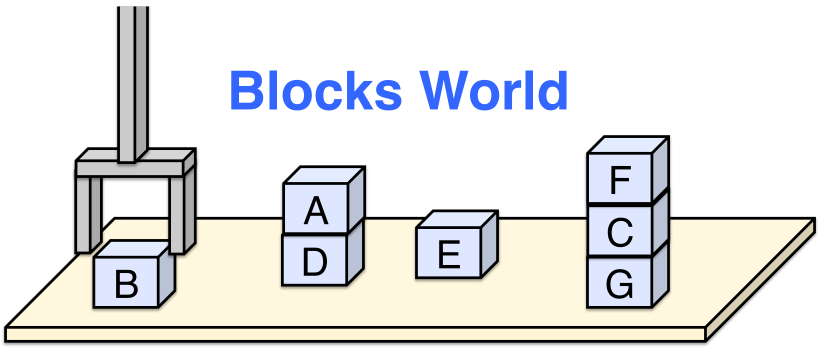
Initial state: s_0 **Set of goal states:** S_g

Task: Given (Σ, s_0, S_g), find a sequence of actions $(a_1, a_2, \ldots, a_{n-1}, a_n)$ that produces a sequence of state transitions $(s_1, s_2, \ldots, s_{n-1}, s_n)$ such that $s_n \in S_g$.

Classical Planning



Blocks World

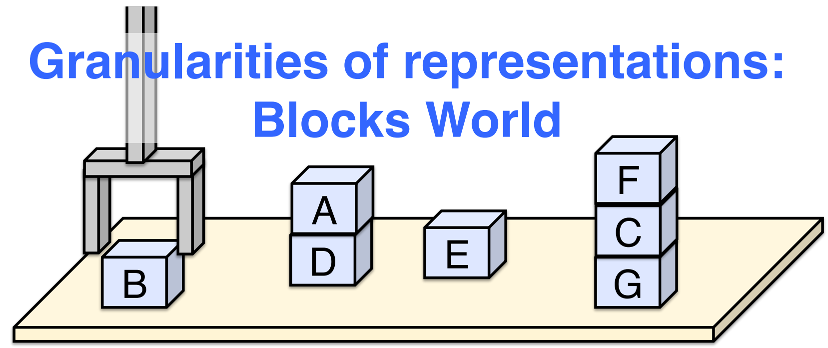


Goals:

- Build a tower of A,B,C,…
- Get block G,
- ….

Silly domain, but concisely illustrates many general planning issues

Granularities of representations: Blocks World



Several ontologies possible
(ways to conceptualize the world and its changes)

Alternative Ontologies: how to move a block

Version 1:
MoveBlock

Version 2:
MoveGripper
GraspBlock
MoveGripper
UngraspBlock

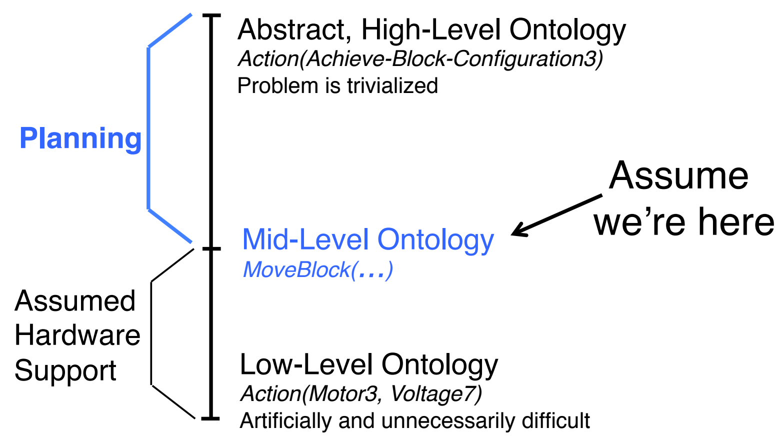
Version 3:
MoveGripper
OpenGripper
MoveGripper
CloseGripper

Version 4:
Motor1 Velocity
Motor2 Velocity
…

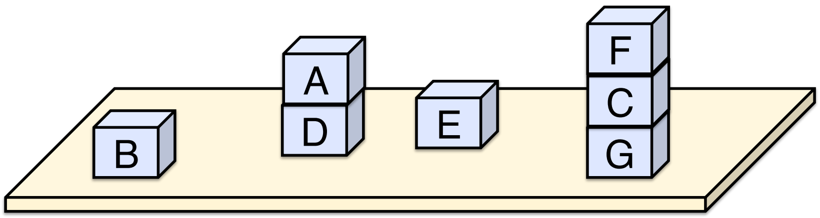
Version 5:
Motor1-Voltage
(current, dutycycle)
…

….

Levels of Ontological Commitment



Traditional Blocks World



Only ***support relationships*** matter (and change):

On(x,y) (x is on y), *Clr(x)* (x is clear)

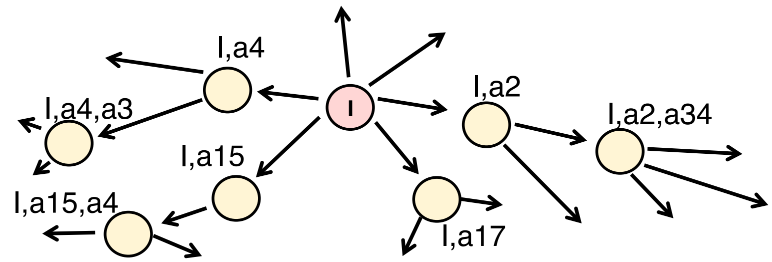
Assumptions:

- A block can support at most one other block
- The table can support any number of blocks
- Generalized block movement *move(x,y,z)*

All reachable situations are defined by…

1) The **initial state**
2) Axioms of world change (= **operator definitions**)

$\Delta \equiv$ Initial State $\cup$ Operator Definitions



Planning as search

Planning is 'just' search:

- Nodes = states
- Edges = actions

But:

- the state space can be **very** large!
- we don't describe states with an atomic label ('s10'), but use a 'factored' representation to capture only the essential properties

Planning vs. Search

| Interesting action sequences | << | All action sequences |

Search operators are "inferentially opaque"

Planning allows reasoning about state features

Planning as theorem proving

Planning: given the initial state, find a sequence of actions that yield a situation where goal holds.
= Use operators to *derive* goal from initial state.

Operator: carry(x)
General knowledge of one kind of action: preconditions and effects

Action: carry(BlockA)
Ground instance of an operator

This week's lectures

Representations for planning:

- Situation calculus
- STRIPS
- PDLL

Planning algorithms:

- Forward chaining and backward chaining
- Propositionalization (SATplan)
- GraphPlan

Representations for planning

Key questions

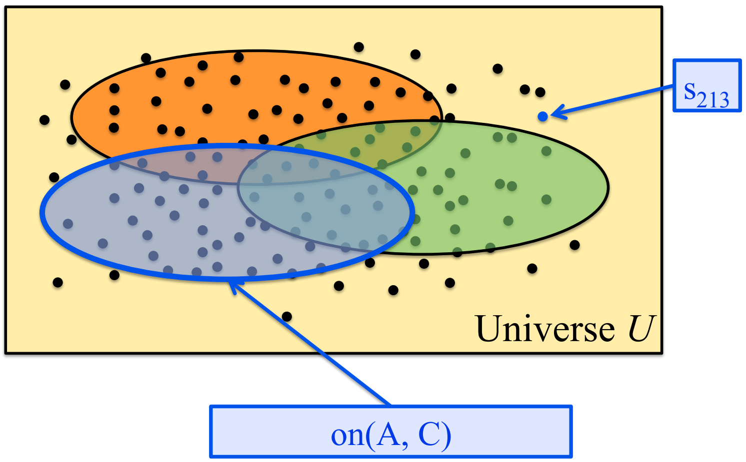
How do we represent states?

- What information do we need to know?
- What information can we (safely) ignore?

How do we represent actions?

- When can we perform an action?
- What changes when we perform an action?
- What stays the same?
- What level of detail do we care about?

Atomic vs. factored state representations



Representations for states

We want to know what state the world is in:

- What are the current properties of the entities?
- What are the current relations between the entities?

Logic representation:
Each state is a conjunction of ground predicates:
$Block(A) \land Block(B) \land Block(C) \land Table(T)$
$\land On(A,B) \land On(B,T) \land On(C,T) \land Clear(A) \land Clear(C)$

Planning: Database semantics

- Each **constant** refers to a **unique object** (no two names for the same object)
- **Domain closure**: the domain consists only of those objects for which we have a name.
- **Closed world assumption**: if we don't know that P is true, we assume it's false.
- **No function symbols**

Fluents vs. atemporal propositions

Atemporal propositions:
Certain properties and relations hold always true: *table(T)* does not change.

Fluents:
on(A, B) may be true in current state, but not after the action *move(A,B,T)* is performed.
on(x, y) is a fluent.
We may add a state variable: *on(x, y, s)*

Representations for operators

The operator definitions model world dynamics:
- When can an action be performed?
- How does the world change as a result?

Different languages:
- Situation Calculus
- Strips Operators
- PDDL Operators*

Pure first-order logic

Specialized syntax

* Ch10 R & N say PDDL but actually discuss Strips

Representations for operators

Operator name (and arity): move x from y to z

move(x,y,z)

Preconditions: when can the action be performed

$clear(x) \land clear(z) \land on(x,y)$

Effects: how does the world change?

$\underbrace{clear(y) \land on(x,z)}_{new} \land \underbrace{clear(x)}_{persist} \land \underbrace{\neg clear(z) \land \neg on(x,y)}_{retract}$

=> main differences between languages

Summary: representations for classical planning

Language: restricted FO predicate logic;
- No functions; database semantics:
- Finite number of predicates and constants

States: sets (conjunction) of ground atoms (positive literals)
- Only finite number of states is possible

Operators: *(name, preconditions, effects)*
- Preconditions and effects: sets of literals

Situation calculus

Operators: Situation Calculus

FOPC with some conventions

Move-Block ontology with:
– at most one block directly on top of another
– a big table (always empty space available)
move(x,y,z) operator: move x from y to z

Fluents require situation variable: *clear(x,s)*

The "Result" Function:

Result: Action × Situation → Situation

State transition function (not a predicate).

Result (Move (A, B, C), S_i*)*

Result (Move (B, Tbl, A), Result (Move (A, B, C), S_i*))*

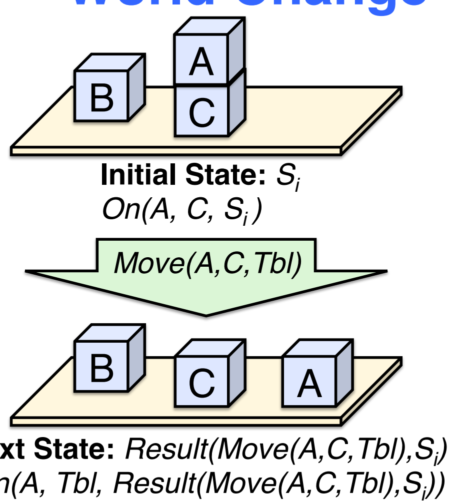
Straightforward generalization to variables:

Result (Move (?x, ?y, C), S_i*)*

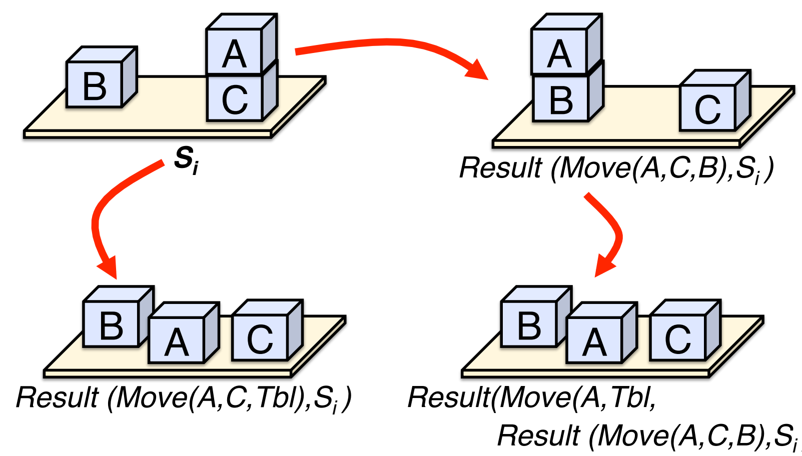
denotes the set of situations where something was just moved to C from the previous state S_i

Useful in "Goal Regression" planning

World Change



Situation calculus identifies states by history, not configuration



Move(x,y,z) definition

$\forall x\ \forall y\ \forall z\ \forall s\ \Theta \Rightarrow \Psi$

If Θ holds in *s*:
- x is on y
- z is clear
- x is a block
- x is clear
- ...

Preconditions Θ

Then Ψ will hold in *Result(Move(x,y,z),s))*:
- x is on z
- y is clear
- ...

Effects Ψ

Move(x, y, z)

NB: this is only a *partial* definition!

$\forall x\ \forall y\ \forall z\ \forall s\ [$

$(Clear(x,s) \wedge Clear(z,s) \wedge On(x,y,s) \wedge Block(x) \wedge Diff(x,z) \wedge Diff(y,z))$

$\Rightarrow$

$(On(x, z, Result(Move(x,y,z), s)) \wedge$

$Clear(y, Result(Move(x,y,z), s)) \wedge$

$Clear(x, Result(Move(x,y,z), s)) \wedge$

$(Table(z) \Rightarrow$
$Clear(z, Result(Move(x,y,z),s)))\ ]$

Conditional Effect

Do we need to retract fluents?

$On(x, y, \mathbf{s})$: situation-specific relation

Do we need to assert **negative** fluents?
$\neg On(x, y, Result\ (Move\ (x, y, z), s))$

No, not in Situation Calculus!

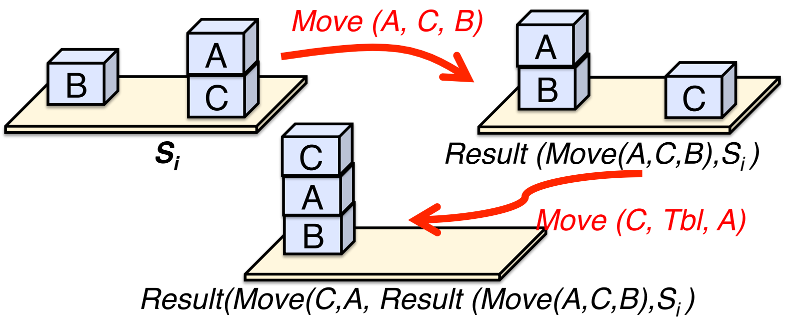
The frame problem

Logic requires an inference path (=derivation) to determine that something holds

Some relations are not affected by an action
We may need to use these relations later

We need to know whether they persist through the action… **(= frame axioms)**
frame = cartoon frame; reference frame (physics)

The need for frame axioms



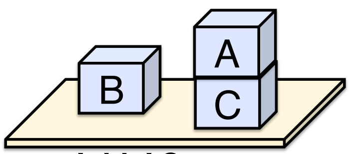
But is this precondition satisfied?

On (C, Tbl, Result (Move (A, C, B), S_i)

And suppose there were other blocks: D, E, F…

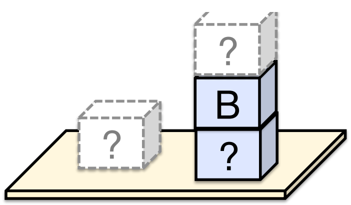
Move(x, y, z) definition extended

$$\forall x\ \forall y\ \forall z\ \forall s\ [$$
$$(Clear(x,s) \wedge Clear(z,s) \wedge On(x,y,s) \wedge Block\ (x) \wedge Diff(x,z))$$
$$\Rightarrow$$
$$([\forall v\ \forall w\ (On(v,w,s)\ \wedge\ Diff(v,x)) \Rightarrow$$
$$On(v,w,Result(Move(x,y,z),s))\ )]$$
$$\wedge\ [\forall v(Clr(v,s)\ \wedge\ Diff(v,z)) \Rightarrow$$
$$Clr(v,\ Result(Move(x,\ y,\ z),\ s))\ )])]$$



Initial State

On(A, C, S_i)
On(C,Tbl, S_i)
On(B,Tbl, S_i)
Blk(A)
Blk(B)
Blk(C)
Table(Tbl)
Clr(A, S_i)
Clr(B, S_i)
Clr(Tbl, S_i)



Goal state *?s*

Find an ?x and *?s* such that:

On(B, ?x, ?s)
Blk(?x)

STRIPS

STRIPS Operators

Observation 1: writing all frame axioms can be very tedious (and error-prone).

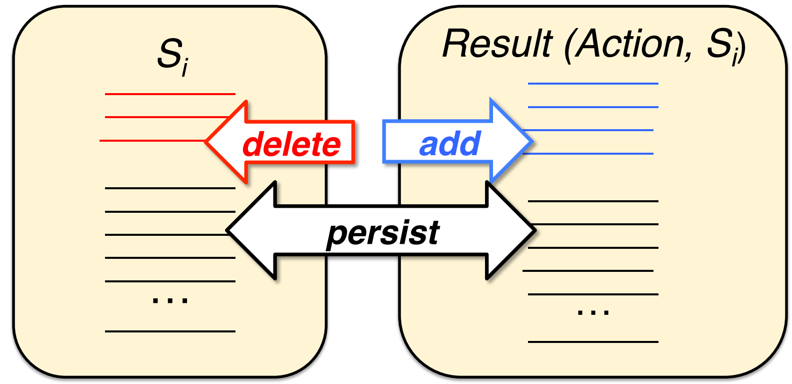
Observation 2: relatively few things change after each action. So just specify what changes

Strips operators are more concise than Situation Calculus

Historically: <u>St</u>anford <u>R</u>esearch <u>I</u>nstitute <u>P</u>roblem <u>S</u>olver

World Changes

Action must fully define resulting world state



Situation Calculus	Strips
Add-set *Persist*-set	*Add*-set *Delete*-set
Fluents are deleted (unless they appear in the *persist*-set)	Fluents persist (unless they appear in the *delete*-set)
No mention = no inference path	More concise, because usually ∣*Persist*-set∣ >> ∣*Delete*-set ∣
Allow conditional effects	Don't allow conditional effects

Strips Operators

Preconditions: a list of positive literals

Effects: two lists of positive literals
- *Delete*-list - things to be retracted
- *Add*-list - things to be asserted

Alternatively, effects can be combined in one list (as in R&N):

Delete elements are designated with "¬"
This is *not* logical negation

Representations

Situation Calculus

Δ contains *all* initial WFFs

No distinction between operators and initial state

Operator definitions distributed throughout Δ

Strips

Operator information centralized, stored separately

State information stored separately for each state

No longer need a situation designator

Closed world assumption

Strips *Move* Operator (?)

Move (x, y, z):

Preconditions:
Clear(x), Clear(z), On(x, y), Block(x), Diff(x, z), Diff(y, z)

Effects: *¬On (x, y), ¬Clr(z), On (x, z), Clr(y)*

What's wrong?

How can we fix it?

We'd like to say something like:

Move(x, y, z):

Preconditions:
Clear(x), Clear(z), On(x, y), Block (x), Diff(x, z), Diff(y, z)

Effects: *¬On(x, y), Block(z) ⇒ ¬Clear(z), On(x, z), Clear(y)*

Now what's wrong?

Two *move* operators

MoveToBlock(x, y, z):
Preconditions: *Clr(x), Clr(z), On(x, y), Blk(x), Blk(z), Diff(x, z), Diff(y, z)*
Effects: *¬On(x, y), ¬Clr(z), On(x, z), Clr(y)*

MoveToTable (x, y, z):
Preconditions: *Clr(x), On(x, y), Blk(x), Tbl(z), Diff(y, z)*
Effects: *¬On (x, y), On (x, z), Clr (y)*

PDLL

PDDL

(Planning Domain Definition Language)

Modern implementation of classical planning language.

Relaxes Strips constraints, allowing negations, Conditional effects, equality, internal quantification, Domain axioms, no Closed World Assumption

Often *implemented* as reduction to Strips

To conclude...

Today's key questions

Planning:

What is it? What's its relation to search and logic?

State and operator representations:

Why are factored state representations better than atomic representations?

What is an operator? How do we represent preconditions and effects? What is the frame problem?

What are the differences between Strips and Situation calculus?

Today’s materials

Much more on planning:

http://www.cs.umd.edu/~nau/planning/slides/

http://planning.cs.uiuc.edu/