Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 and MoSIG M1 — Winter Semester 2013
Lecture 3 — 15 February 2013

Planning as Search: BlocksWorld

The Intelligent Agent 2

Planning and Problem Solving.. 2

Blocks World 4

Planning as Search : 8

The Intelligent Agent

To provide a formal basis for studying intelligence, Nils Nilsson has proposed the Intelligent Agent as a fundamental concept for formalizing intelligence.

The Intelligent Agent has 3 components: (A, B, C)
A) Actions; The ability to act; A physical body;
B) Goals. (In French "Buts")
C) Knowledge; The ability to choose actions to accomplish goals.

The "Intelligent Agent" acts based on the principle of Rationality.

Rational behavior: Actions are chosen to accomplish goals

Nilsson proposed to define intelligence as rationality:

Rational Intelligence: The ability to choose actions to accomplish goals.

An agent is intelligent if it 1) can act, 2) has goals, and 3) Can choose its actions to accomplish it's goals.

Rational intelligence leads to a formulation of intelligence as problem solving and planning.

Planning and Problem Solving

Planning: The search for a sequence of actions leading to a goal.

Rationality leads to a formulation of intelligence as planning
Rational intelligence is formalized using a Problem space.

A problem space is defined as
1) A set of states {U},
2) A set of operators for changing states {A} (Actions).

A problem is {U}, {A} plus
an initial state $i \in \{U\}$
a set of Goal States $\{G\} \subset \{U\}$

A plan creates a sequence of actions A_1, A_2, A_3, A_4,... that lead from the state S to one of the states $g \in \{G\}$

States: A state, s, is a "partial" description of the real universe.

A state is defined as a conjunction of predicates (Truth functions) based on measured (observed) values. The measured values are called "observations".

Examples:

Mobile Robotics: Near(x, y, t)

Blocks World: OnTable (A) $\wedge$ On (A, B) $\wedge$ HandEmpty

Blocks World

Blocks world is an abstract, toy world for exploring problems. Blocks world is a "Closed" world. It has a finite number of states.

Blocks world is composed of a finite number of blocks in a finite number of states.

Blocks world is composed of:

- A set of blocks
- An agent that can act on blocks to change their state

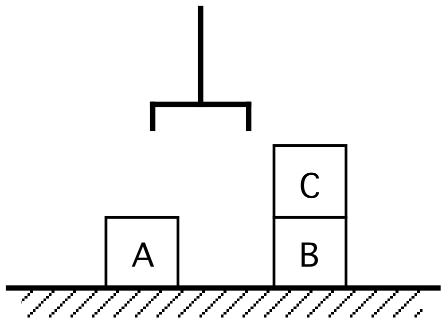
Classic Definitions:
1) A universe composed of a set of cubic blocks and a table
2) Blocks are mobile, the table is immobile
3) The agent is a mobile hand,
4) A block can sit on a table, on another block, or in the hand.
5) There cannot be more than one block on another block
6) The table is large enough for all blocks to be on the table.
7) The hand can move only one block at a time.

The state of the universe is formalized using first order predicates.
Blocks are represented by Capital Letters {A, B, C, …}
Variables (lower case letters) can represent sets of blocks
This are specified by Quantizers: for-all x ($\forall$x:), There-exists x: ($\exists$y:)

Predicates:

On(x, y)	Block x is on Block y.	
OnTable(x)	Block x is on the Table.	
Held(x)	Block x is in the hand.	
Free(x)	F(x)	No block is On x : $\neg\exists$y: (On(y, x)) or $\forall$y: ($\neg$On(y, x))
HandFree	The hand is empty, or $\neg\exists$x (H(x))	

For example:



HandFree ∧ OnTable(A) ∧ OnTable(B) ∧ On(C, B) ∧ Free(C) ∧ Free (A)

Actions:

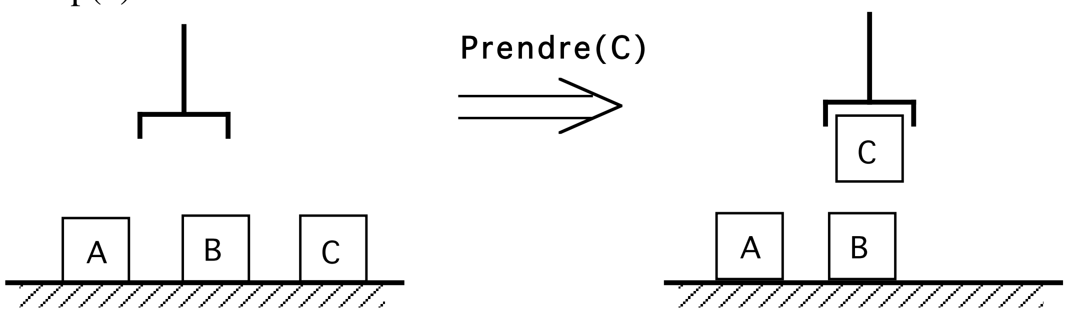
Actions are state change operators. Actions are atomic.
Nilsson proposed to formalize actions with STRIPS :
(Stanford Research Institute Problem Solver) (1971).

Principle: explicitly list all state changes.

Action: Name(Variables)

Precondition: Must be true for the action to operate
Retract: rendered false by the action
Add: rendered true by the action.

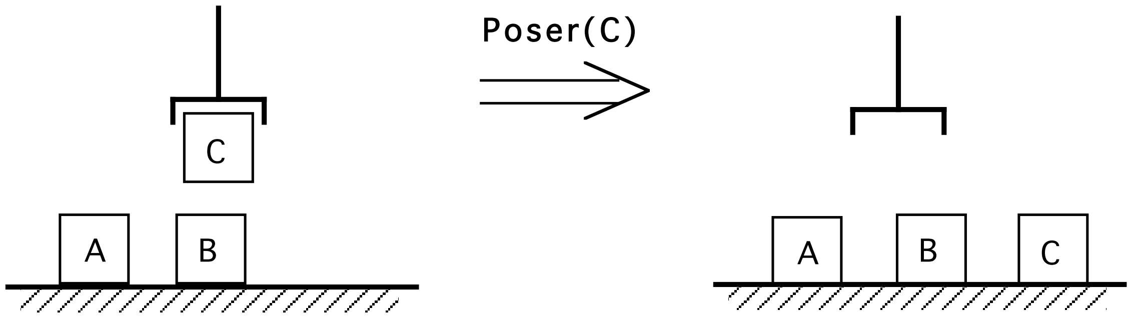
Grasp(x)



Grasp(x)

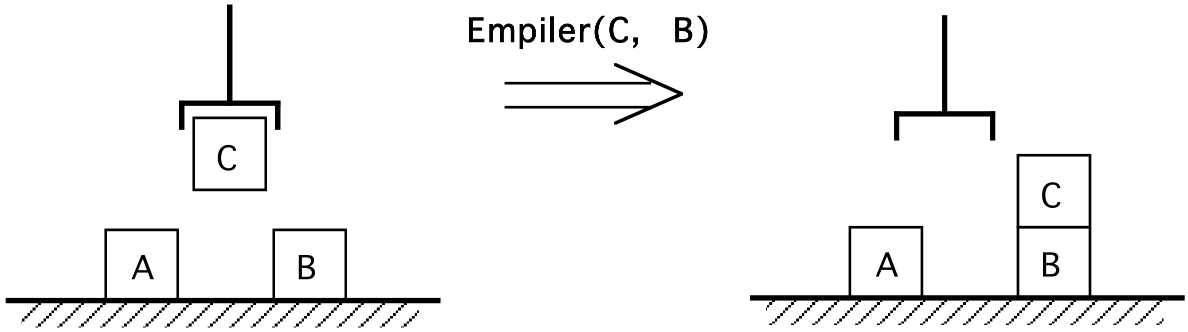
Precondition: HandFree ∧ Free(x) ∧ OnTable(x)
Retract: HandFree ∧ Free(x) ∧ OnTable(x)
Add: Held(x)

Pose(x)



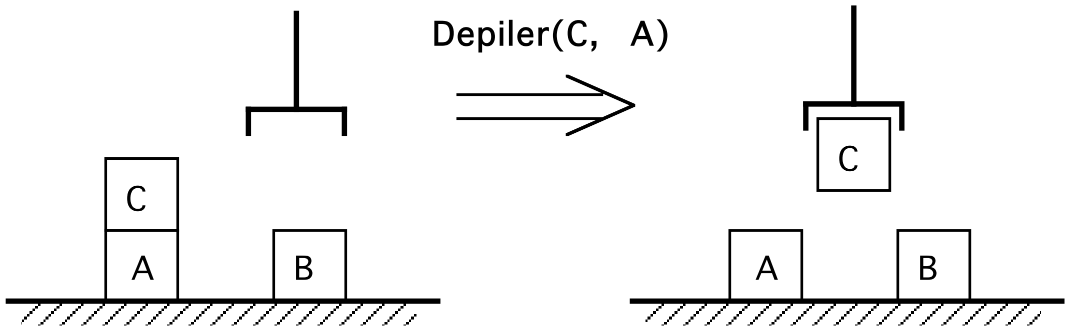
Pose (x)
Precondition: Held(x)
Retract: Held(x)
Add: HandFree ∧ Free(x) ∧ OnTable(x)

Stack(x, y)



Stack (x, y)
Precondition: Held(x) ∧ Free(y)
Retract: Held(x) ∧ Free(y)
Add: Free(x) ∧ On(x, y) ∧ HandFree

Unstack (x, y)



Unstack(x, y)
Precondition: Free(x) ∧ On(x, y) ∧ HandFree
Retract: Free(x) ∧ On(x, y) ∧ HandFree
Add: Held(x) ∧ Free(y)

Question: Why do we need Pose(x). Is not Stack(x, table) equivalent?

Response: If we execute Stack(x, table) the predicate Free(table) is not true.

Planning as Search

A problem is defined by
a universe, {U},
an initial state, i
A set of Goal states, {G}.

Planning is the generation of a sequence of actions to transform i to a state g ∈ {G}

The "paradigm" for planning is "Generate and Test".

Given a current state, s
1) Generate all neighbor states {N} reachable via 1 action.
2) For each n ∈ {N} test if n ∈ {G}. If yes, exit
3) Select a next state, s ∈ {N} and loop.

Planning requires search over a graph for a path.

A taxonomy of graph search algorithms includes the following

1) Depth first search
2) Breadth first search
3) Heuristic Search
4) Hierarchical Search

The first three are unified within the GRAPHSEARCH algorithm of Nilsson.

Graph searching has exponential algorithm complexity.
"knowledge" can be used to reduce the complexity.