Methods of Artificial Intelligence

Kai-Uwe Kühnberger & Nico Potyka

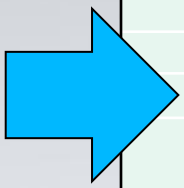
Planning

UNIVERSITÄT OSNABRÜCK

Organization: 27th and 28th of November

- On the 27th and 28th of November I and Nico Potyka will be busy with the hiring committee for the professorship “Cognitive Natural Language Processing and Communication”
- We plan the following:
  - On 27th of November: we would ask you to attend at least one scientific talk and one lecture of a candidate for the professorship (Schedule: see next slide)
  - On the 28th of November: Lucas Bechberger will present work on conceptual spaces
- Important: the talks of the candidates for the professorship will not be relevant for the exams.
- Lucas Bechberger’s presentation on conceptual spaces will be relevant for exams.

Tentative Schedule



1.	23.10.2017	Introduction
	Part I: Foundations & Planning	
2.	30.10.2017	Local Search
3.	06.11.2017	Constraint Satisfaction Problems Advanced
4.	13.11.2017	Theorem Proving Advanced
5.	20.11.2017	Planning
	Part II: Reasoning & Learning	
6.	27.11.2017	Knowledge Representation
7.	05.12.2017	Midterm
8.	11.12.2017	Reasoning over Space and Time
	18.12.2017	Uncertain Reasoning and Learning Basics
9.	08.01.2018	ML 1: SVMs and Random Forests
10.	15.01.2018	ML 2: Reinforcement Learning
	Part III: Cognitive Architectures & Games	
11.	22.01.2018	Cognitive Architectures
12.	29.01.2018	Games Advanced
13.	05.01.2018	Repetition
14.	06.02.2018	Final Exam

This Week

Today: Classical Planning Accounts

- General Remarks / The Frame Problem
- Deductive Planning Systems: The Situation Calculus
  - Deductions
  - Frame Axioms
- State-Based Planning: STRIPS
  - Definition
  - Examples and Evaluation
- Miscellaneous
  - Linear Planning vs Non-Linear Planning
  - A Note on Real-World Planning and Behavior-Based Robotics
- Newer Developments in Planning
  - Hierarchical Task Networks (HTNs) & Skill Learning

Tomorrow: Markov Decision Processes

Planning Systems

General Remarks

AI and Planning

Meaning of planning in AI:

- AI planning deals with the formalization, implementation, and evaluation of algorithms for constructing plans.

What is a plan?

- A plan is a sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

Two types of planning:

- Basic approach to state-based planning: Strips.
- Basic approach to deductive planning: Situation Calculus.

Another alternative to planning:

- Markov Decision Processes / Reinforcement learning of action sequences (see tomorrow).

Components of Planning Systems

A language to represent states, goals, and operators is needed.

- Typically, different subsets of FOL for states, goals and operators.
- States are usually considered as conjunctions of positive ground literals, goals might be more complex (allowing quantified variables, disjunction, etc.).
- In contrast to problem solving, operators are typically represented as schemes.
- Operator semantics: achieving a state change by applying an action (state space model).

An algorithm for constructing a plan

- A search algorithm
- In contrast to problem solving, typically backwards.

A Story about the Frame Problem

"Once upon a time there was a robot, named R1 by its creators. Its only task was to fend for itself. One day its designers arranged for it to learn that its spare battery, its precious energy supply, was locked in a room with a time bomb set to go off soon. […]"

"There was a wagon in the room, and the battery was on the wagon, and R1 hypothesized that a certain action which it called PULLOUT(WAGON,ROOM) would result in the battery being removed from the room. Straightaway it acted, and did succeed in getting the battery out of the room before the bomb went off. Unfortunately, however, the bomb was also on the wagon. R1 knew that the bomb was on the wagon in the room, but didn't realize that pulling the wagon would bring the bomb out along with the battery. Poor R1 had missed that obvious implication of its planned act."

A Story about the Frame Problem (cont.)

"[…] 'The solution is obvious,' said the designers. 'Our next robot must be made to recognize not just the intended implications of its acts, but also the implications about their side-effects, by deducing these implications from the descriptions it uses in formulating its plans.' They called their next model, the robot-deducer, R1D1."

"They placed R1D1 in much the same predicament that R1 had succumbed to, and as it too hit upon the idea of PULLOUT(WAGON,ROOM) it began, as designed, to consider the implications of such a course of action. It had just finished deducing that pulling the wagon out of the room would not change the color of the room's walls, and was embarking on a proof of the further implication that pulling the wagon out would cause its wheels to turn more revolutions than there were wheels on the wagon - when the bomb exploded."

(Dennett, 1987, pp. 41-2)

Summary of Classical Problems

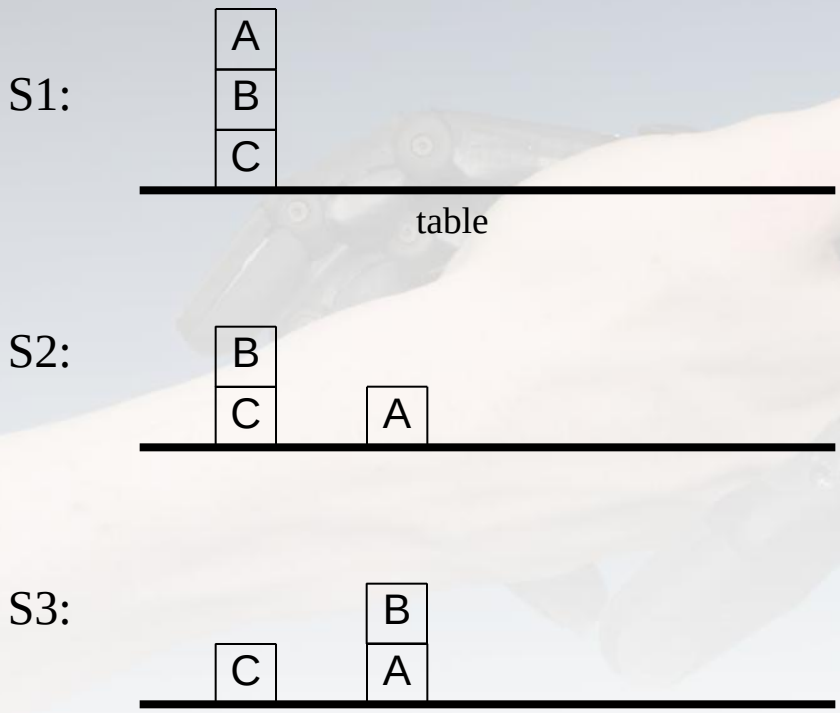
There are three famous problems in the context of planning systems.

- Frame problem:
  - Usually actions change several facts about the world, but most facts are not changed.
    - In the real world, almost everything stays the same relative to one single action.
- Qualification problem:
  - A true description about real actions requires an endless list of preconditions.
    - What is relevant for a particular action?
- Ramification problem:
  - Real actions have many secondary consequences.
    - How can this be handled?

The frame problem (and its variations) is (are) probably one of the most discussed problems in AI-related philosophy.

In particular, in the context of behavior-based AI, evolutionary robotics etc. such problems are strongly discussed.

Simple Planning Domain: Blocksworld



- A (minimal) blocksworld consists of the following entities:
  - Objects:
    - Table
    - Set of blocks: {A,B,C}
  - Predicates: {on(block,block),ontable(block) clear(block)}
  - Actions:

    {put(block,object,state), puttable(block,state)}

Deductive Planning Systems

The Situation Calculus

Deductive Planning

The Situation Calculus

History:

- Situation calculus: Introduced by McCarthy (1963) and used for plan construction by resolution by Green (1969).
- Further developments in the context of the event calculus (Kowalski, 1986) and Fluent calculus (Thielscher, 2000).

Features:

- Deductive inferences are used to solve planning problems.
- In general: Situation calculus is an extension of FOL.
  - There are connections to modal logic.
- Idea: Prove logically that a set of goals follows from an initial state given operator definitions (axioms).
- Perform the proof in a constructive way.
  - Plan is constructed as a byproduct of the proof.

Deductive Planning

Here is a simple example of a proof in the situation calculus

Axioms:

A1 on(a,table,s1) (Literal of the initial state)

A2 $\forall S[on(a,table,S) \rightarrow on(a,b,put(a,b,S))] \equiv$ (Axiom)

$\neg on(a,table,S) \vee on(a,b,put(a,b,S))$ (Clausal form)

Proof the goal predicate on(a,b,G)

1. $\neg on(a,b,G)$		(Negation of the theorem)
2. $\neg on(a,table,S) \vee on(a,b,put(a,b,S))$		(A2)
3. $\neg on(a,table,S)$	$G \mapsto put(a,b,S)$	(Resolve 1,2)
4. on(a,table,s1)		(A1)
5. contradiction	$S \mapsto s1$	(Resolve 3,4)

s1 with on(a,table,s1) is true and on(a,b,s2) holds. s2 can be reached by putting a on b in situation s1: s2 = put(a,b,s1).

Frame Problem Revisited

Notice:

- No closed world assumption is assumed, hence we have the full expressive power of FOL.

Problem:

- Additionally to axioms describing the effects of actions, frame axioms become necessary (compare remarks on the frame problem).
- Frame axioms specify the invariants of the domain: relations that remain unaffected by the performance of an action.
- Frame axioms are necessary to allow proving conjunctions of goal literals.

Example of a frame axiom:

- $\forall S[on(Y, Z, S) \rightarrow on(Y, Z, put(X, Y, S))]$
  - After a block X was put on a block Y, it still holds that Y is placed on a block Z, if this did hold before the action was performed.

Blocks World in Prolog

Remark: In the following different variables refer to different entities. This must be made explicit in most logical formalisms

Effect Axioms

on(X,Y,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S)
clear(Z,put(X,Y,S))	←	on(X,Z,S) ∧ clear(X,S) ∧ clear(Y,S)
clear(Y,puttable(X,S))	←	on(X,Y,S) ∧ clear(X,S)
ontable(X,puttable(X,S))	←	clear(X,S)

Frame Axioms

clear(X,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S)
clear(Z,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S) ∧ clear(Z,S)
ontable(Y,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S) ∧ ontable(Y,S)
ontable(Z,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S) ∧ ontable(Z,S)
on(Y,Z,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S) ∧ on(Y,Z,S)
on(W,Z,put(X,Y,S))	←	clear(X,S) ∧ clear(Y,S) ∧ on(W,Z,S)

Blocks World in Prolog

Frame Axioms continued

clear(Z,puttable(X,S))	$\leftarrow$ clear(X,S) $\wedge$ clear(Z,S)
ontable(Z,puttable(X,S))	$\leftarrow$ clear(X,S) $\wedge$ ontable(Z,S)
on(Y,Z,puttable(X,S))	$\leftarrow$ clear(X,S) $\wedge$ on(Y,Z,S)
clear(Z,puttable(X,S))	$\leftarrow$ on(X,Y,S) $\wedge$ clear(X,S) $\wedge$ clear(Z,S)
ontable(Z,puttable(X,S))	$\leftarrow$ on(X,Y,S) $\wedge$ clear(X,S) $\wedge$ ontable(Z,S)
on(W,Z,puttable(X,S))	$\leftarrow$ on(X,Y,S) $\wedge$ clear(X,S) $\wedge$ on(W,Z,S)

Example of a planning problem

Facts (Initial state)

on(d,c,s_1)	on(c,a,s_1)
clear(d,s_1)	clear(b,s_1)
ontable(a,s_1)	ontable(b,s_1)

Theorem (Goal)

on(a,b,G) $\wedge$ on(b,c,G)

Remarks

Fluents:

The concept fluent denotes a relation for which its truth value may vary from state to state.

Frame Axioms:

Although frame axioms are a general solution to the frame problem, the number of frame axioms can be significantly large (resulting in complexity problems).

The three major problems in planning approaches are: the frame problem, the qualification problem, and the ramification problem.

Qualification problem: What needs to be true for a particular action to be possible?

Precondition axioms are a possibility to solve the qualification problem.

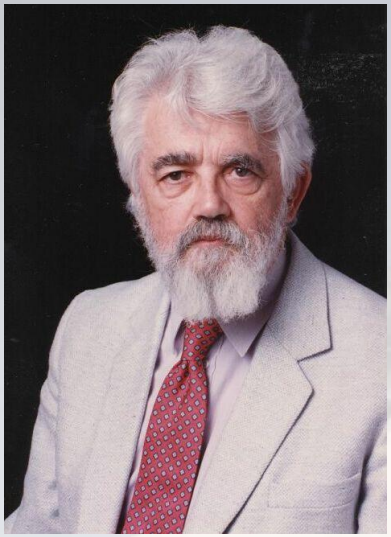
Ramification problem: How can we specify *all* (and even implicit) consequences of an action?

Certain rules can solve the ramification problem (”if an object is attached to another object and one of the objects is moved, the other object moves, too”).

Frame problem: What does *not* change after an action?

Frame axioms are a possibility to solve the frame problem.

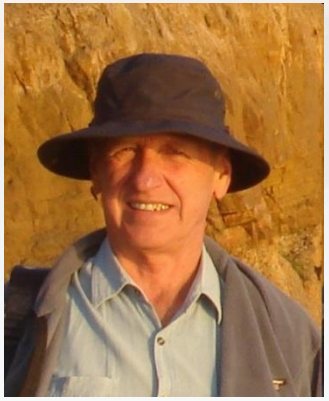
People



John McCarthy

Daniel Dennett

Richard Fikes



Robert Kowalski

Gerry Sussman

State-Based Planning: STRIPS

"STanford Research Institute Problem Solver"

Specification of STRIPS

State descriptions are conjunctions of positive ground literals with domain objects (constants).

D = {A,B,C,D}

P = {ontable^1, clear^1, on^2}

{ontable(A), ontable(C), ontable(D), clear(A), clear(B), clear(D), on(B,C)}

Only conjunctions of literals occur: write them as a set of literals.

Goals: Typically partial state descriptions, e.g. G = {on(B,C), ontable(C)}.

A state s is called a goal state if $G \subseteq s$.

Further features:

CWA: All relations not given explicitly or derivable are assumed to be false.

A state description corresponds to a state in a state-space model.

Typically, only “relevant” aspects of a state are represented.

Strips Operator Schemes

Idea:

An operator consists of a precondition (PRE) and an effect.

Effects are represented as ADD and DEL lists (more precisely sets).

In the most simple case, PRE, ADD, DEL are conjunctions of literals.

Variables are assumed to be existentially quantified, literals in the precondition are positive.

Operator *put:*

Operator:	put(?x,?y)
PRE:	{ontable(?x), clear(?x), clear(?y)}
ADD:	{on(?x,?y)}
DEL:	{ontable(?x), clear(?y)}

Operator Application

Precondition:

- The precondition of an operator is matched against the current state.
- If $PRE_\sigma \subseteq s$ then the precondition holds and the operator can be applied.

Substitution:

- Substitution σ is applied on the complete operator scheme, i.e. the variables occurring in DEL and ADD are simultaneously instantiated.
- Strips is propositional (no quantification), that is, variables are always replaced by constant symbols.
- Instantiated operators are called actions.

Instantiated Operators

The put operator once more again:

Operator:	put(A,B)
PRE:	{ontable(A), clear(A), clear(B)}
ADD:	{on(A, B)}
DEL:	{ontable(A), clear(B)}

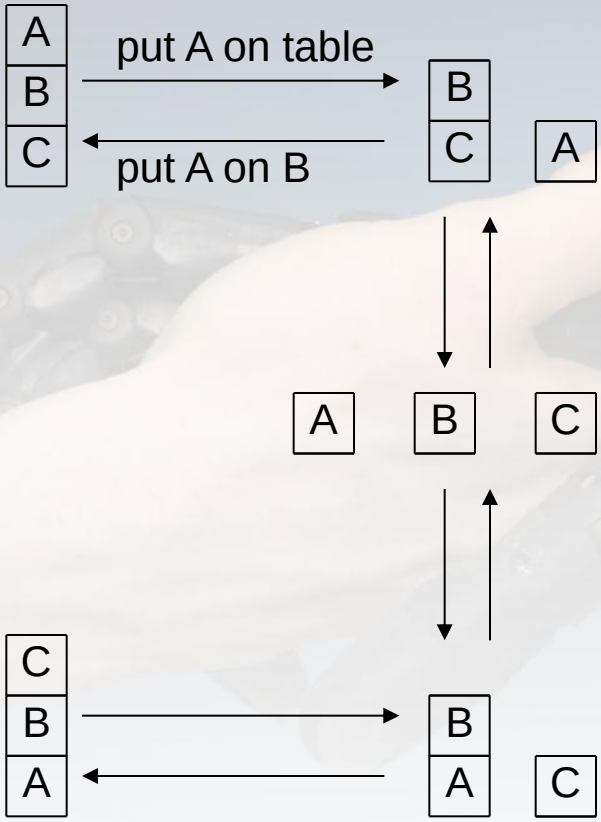
$\sigma = \{x \leftarrow A, y \leftarrow B\}$

With *o* we denote a fully instantiated operator, with PRE(*o*), ADD(*o*), DEL(*o*) the according component of *o.*

Operator application is defined as:

$$Result(o,s) = [s \setminus DEL(o)] \cup ADD(o) \quad \text{if} \quad PRE(o) \subseteq s$$

Example: State-Space Model



Arc_j from a state s_i to a state s_j if and only if state s_j can be reached from s_i by performing a single action

Blocksworld Domain: Operators

The put operator in two different situations:

put(?x, ?y)

PRE: {ontable(?x), clear(?x),clear(?y)}

ADD: {on(?x, ?y)}

DEL: {ontable(?x), clear(?y)}

put(?x, ?y)

PRE: {on(?x, ?z), clear(?x), clear(?y)}

ADD: {on(?x, ?y), clear(?z)}

DEL: {on(?x, ?z), clear(?y)}

The puttable operator:

puttable(?x)

PRE: {clear(?x), on(?x, ?y)}

ADD: {ontable(?x), clear(?y)}

DEL: {on(?x, ?y)}

Example of a Blocksworld Problem:

Blocksworld domain +

GOAL: {on(A,B), on(B,C)}

INITIAL STATE:

{on(D, C), on(C, A), clear(D), clear(B), ontable(A), ontable(B)}

Planning Algorithms

Typically: variants of depth-first search.

- Planning domains are usually too complex for applying breadth-first search.

Forward planning:

- Start with the initial state, built a search tree by transforming a given state by action application, check for cycles, backtrack if you reach a dead-end, terminate if a goal state is reached or if all paths are generated (for finite domains).

Backward planning:

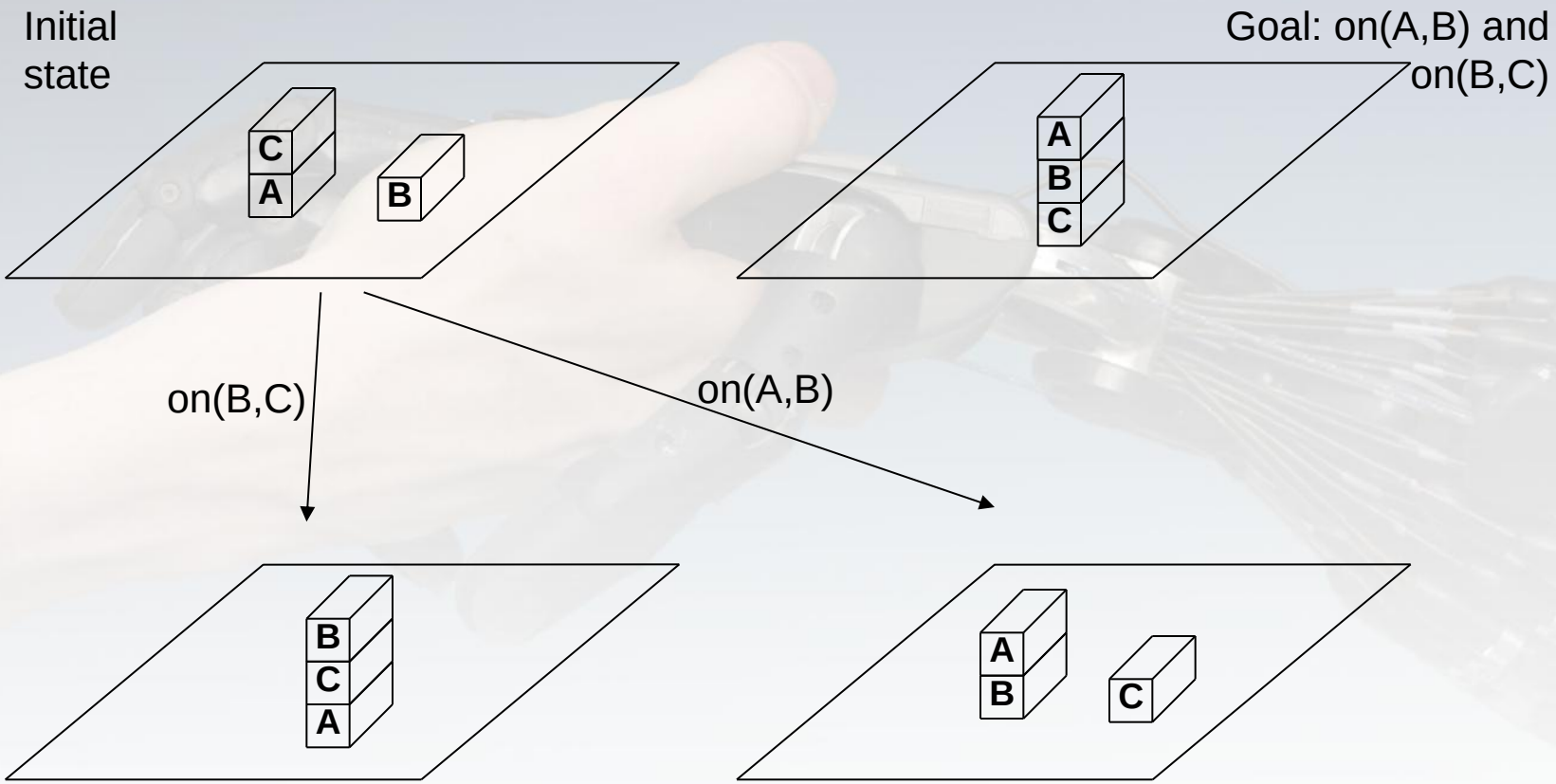
- Start with the top-level goals (a partial description of the goal state), build a search tree by transforming a given state by backwards action application, ..., terminate if a state is reached which subsumes the initial state or ...

Miscellaneous

Linear Planning vs. Non-Linear Planning

A Note on Real-World Planning

Sussman Anomaly



Incompleteness of Linear Planners

Linear planning corresponds to dealing with goals organized in a stack:

[*on*(*A,B*), *on*(*B,C*)]

try to satisfy goal *on(A,B)*

solve sub-goals *[clear(A), clear(B)]*[a]

all sub-goals hold after *puttable(C)*

apply *put(A,B)*

goal *on(A,B)* is reached

try to satisfy goal *on(B,C)*

---

[a] We ignore the additional subgoals ontable(A) resp. on(A,Z) here.

Interleaving Goals

Non-linear planning:

- Non-linear planning allows that a sequence of planning steps dealing with one goal is interrupted to deal with another goal.
  - For the Sussman Anomaly, that means that after block C is put on the table pursuing goal *on(A, B)*, the planner switches to the goal *on(B, C).*
- Non-linear planning corresponds to dealing with goals organized in a set.
- The correct sequence of goals might not be found immediately but might involve backtracking.

A solution for the Sussman anomaly:

{on(A,B), on(B,C)}

try to satisfy goal *on(A,B)*

{clear(A), clear(B), on(A,B), on(B,C)}

clear(A) and *clear(B)* hold after *puttable(C)*

try to satisfy goal *on(B,C)*

apply *put(B,C)*

try to satisfy goal *on(A,B)*

apply *put(A,B)*.

Evaluation of Planners

Termination

Critical case: no solution exists.

Soundness

If every plan returned is a legal sequence of actions to achieve the goal, then the plan is consistent: each intermediate state appearing in the plan is a legal state of the domain.

Completeness

The planner finds a solution, if one exists.

Optimality

The returned plans are optimal (shortest) solutions (typically not considered).

Others

Expressiveness of the planning language.

Efficiency.

Planning Vocabulary

Total vs. partial-order planners:

- Returned plan is a totally ordered sequence of actions vs. some independent actions are given in parallel.

Linear vs. non-linear planners:

- Linear planners consider a goal at a time and work with a goal-stack, non-linear planners allow interleaving of goals.
- All modern planners are non-linear.

Planning Domain:

- Operator Schemes (for extended formalisms additionally types, axioms, functions, ...).
- General description of a domain, such as blocksworld.

Planning Problem and Languages

Planning Problem:

- A domain, an initial state and a planning goal.
- The problem, not the domain, constitutes a set of domain objects.

PDDL:

- Standardized, extended Strips language for state-based planners: PDDL (Planning Domain Definition Language), see, e.g. http://www.icaps-conference.org/index.php/Main/Competitions

Languages:

- ADL (Action Description Language) is an extension of STRIPS by adding functions, an equality predicate, types etc.
- PDDL includes sublanguages for ADL and STRIPS.
  - It has a Lisp-based syntax. Classically state-based planning is realized in Lisp. Today most planners are in C.

A Note on Real-World Planners

Planners intended to work in the real world need to have some more features than ADL (action description language) or STRIPS have.

Here are some of these features (most of them are reflected in PDDL):

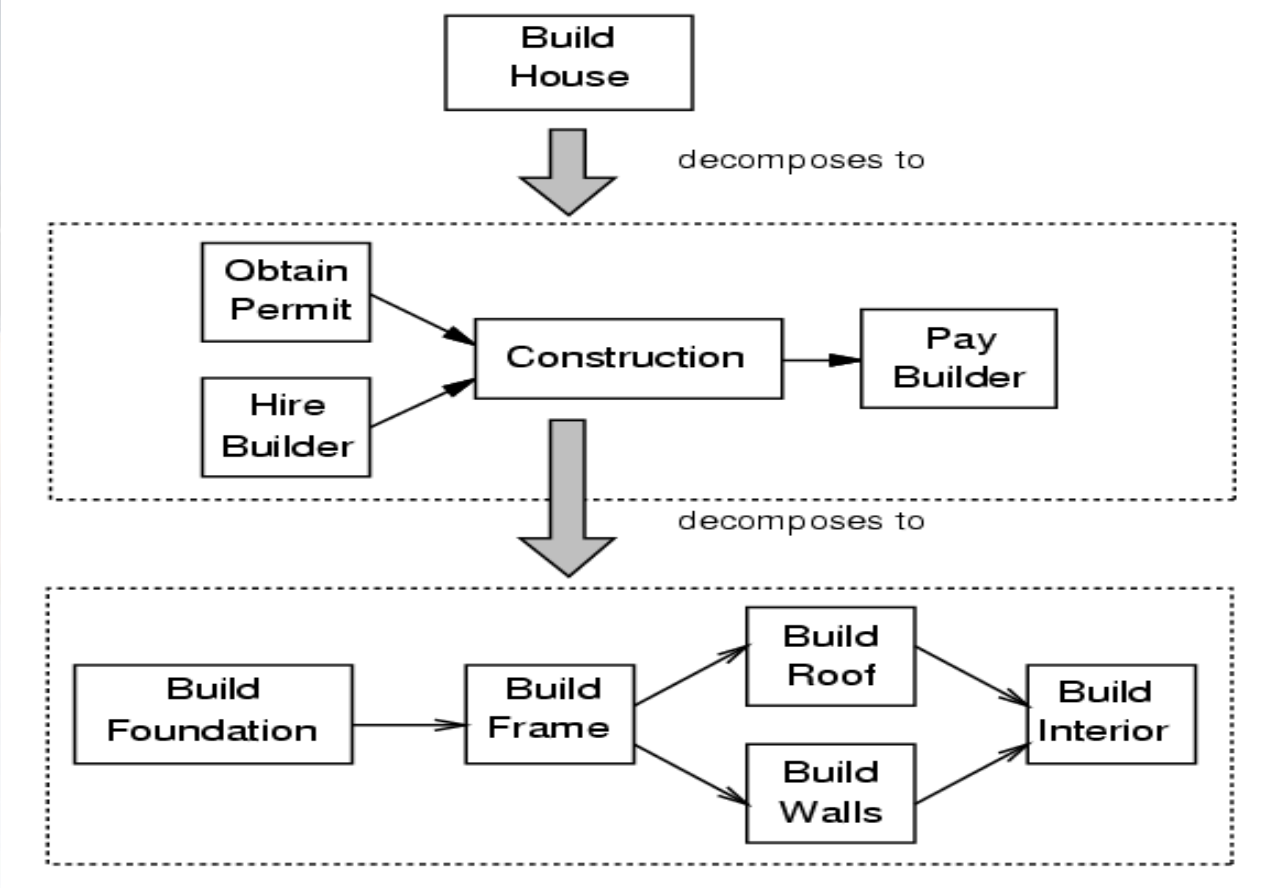
- Time constraints.
  - When does an action start, when does it end?
- Constraints on resources.
  - If only one hoist is available to lift a motor, then no two motors can simultaneously be lifted.
  - The resource requirement is a temporary effect and a prerequisite.
- Incomplete and incorrect information.
  - Incompleteness: the model of the world is usually partial.
    - Belief search.
  - Incorrectness: Model of the world does not necessarily fit to the world.
    - Some form of execution monitoring is necessary.
- Cooperation, competition, coordination etc. with other agents.
  - Building models of models of other agents etc.

Newer Developments in Planning

Hierarchical Task Networks
Skill Learning

Hierarchical Task Networks (HTNs)

- A further development concerns so-called hierarchical task networks.
- Idea: Use subplans in a planning problem and knowledge about your domain in order to decompose tasks.



Hierarchical Task Networks (HTNs)

In the following example taken from a presentation of Langley et al.:

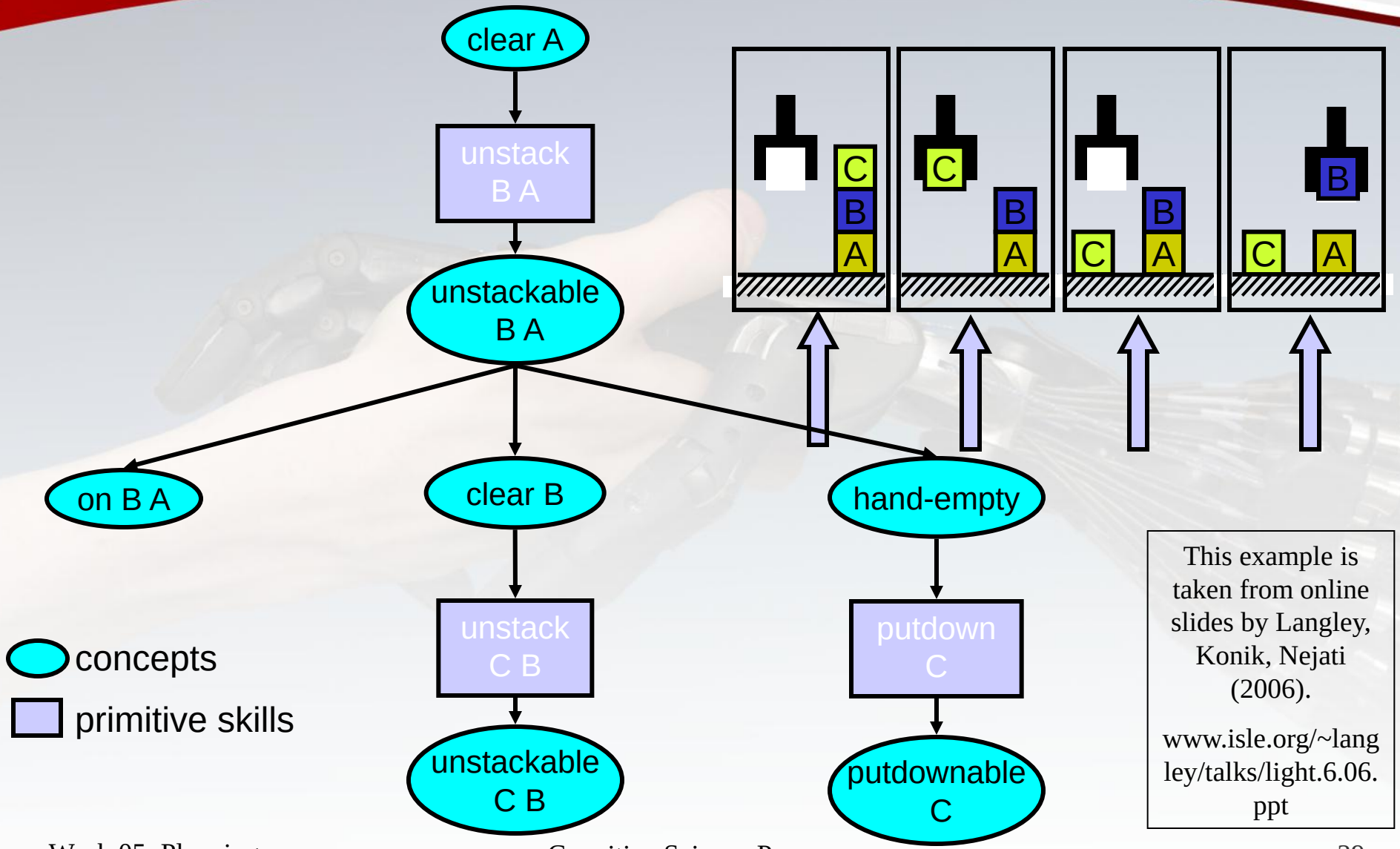
Given:

- A set of domain operators with known effects
- A worked out problem solution that consists of
  - The goal to be achieved in the problem
  - A sequence of operator instances that achieves the goal
  - A related sequence of intermediate problem states

Find:

- A hierarchical task network that
  - Reproduces the solution to the training problem
  - Generalizes well to related problems in the domain

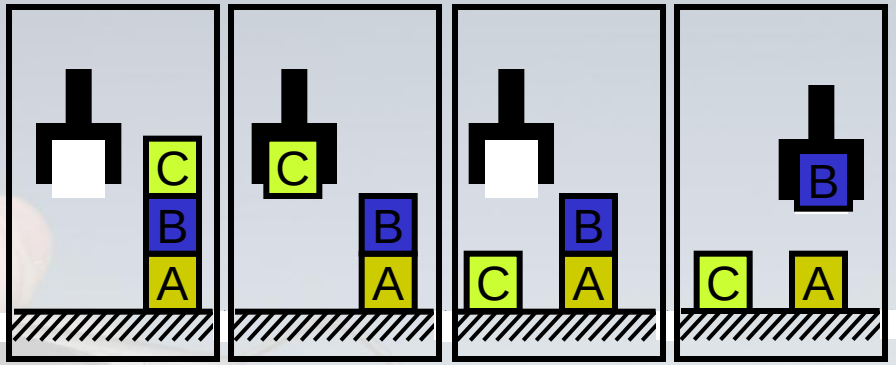
Constructing an Explanation

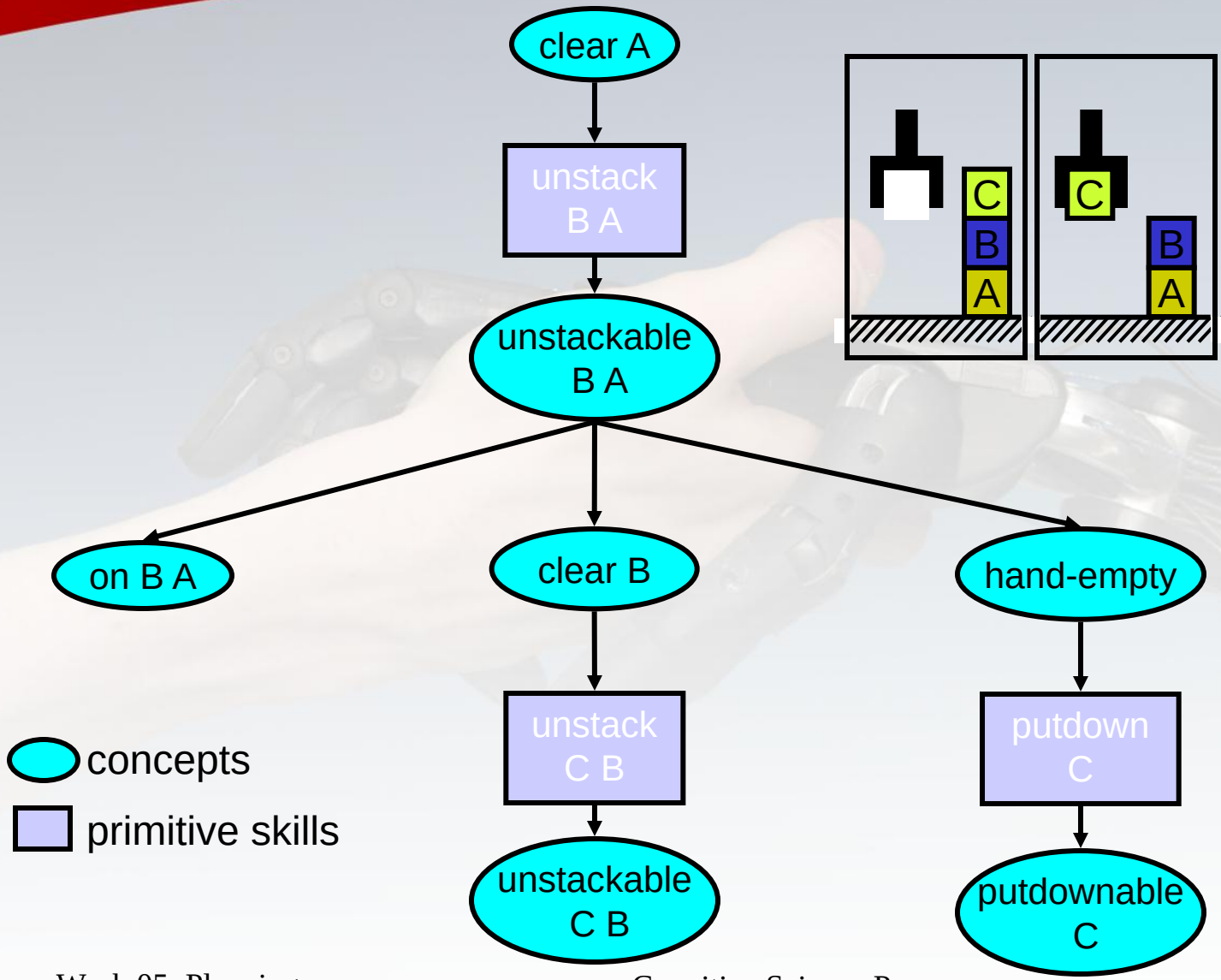


This example is taken from online slides by Langley, Konik, Nejati (2006).

www.isle.org/~langley/talks/light.6.06.ppt

From an Explanation to an HTN

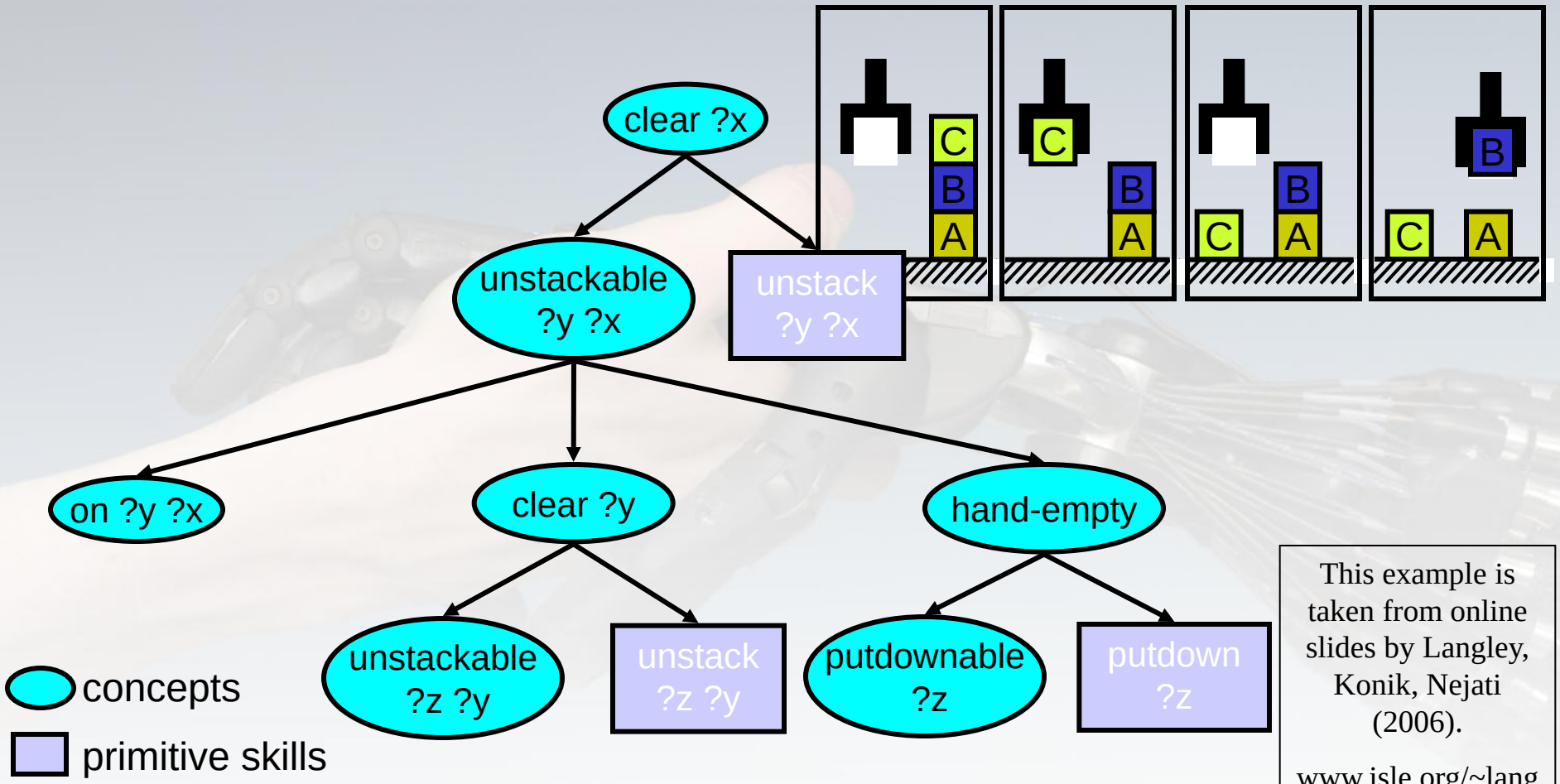




This example is taken from online slides by Langley, Konik, Nejati (2006).

www.isle.org/~langley/talks/light.6.06.ppt

From an Explanation to an HTN



This example is taken from online slides by Langley, Konik, Nejati (2006).

www.isle.org/~langley/talks/light.6.06.ppt

References

Situation Calculus

- Thielscher, M. (2000): The Concurrent, Continuous Fluent Calculus, Studia Logica 67(3):315-331.
- McCarthy, J. (1963). Situations, actions, and causal laws. Technical report, Stanford University.
- Reiter, R. (1991). The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz (ed.), *Artificial intelligence and mathematical theory of computation: papers in honour of John McCarthy*, pages 359–380, San Diego, CA, USA. Academic Press Professional, Inc. 1991.
- Green, C. (1969). Theorem proving by resolution as a basis for question-answering systems. In Meltzer, B. & Michie, D., eds.: Machine Intelligence 4:183–205.

Philosophy

- Daniel Dennett, D. (1987), “Cognitive wheels: The frame problem of AI," in C. Hookway (Ed.), Minds, machines and evolution, Baen Books, 1987.

References

Strips and HTN

- Fikes, R. & Nilsson, N. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2:189-208.
- Ghallab, M., Nau, D., Traverso P. (2004). Automated Planning Theory and Practice. Morgan Kaufmann.

Reinforcement Learning

- Sutton, R. & Barto, A. (1998). Reinforcement Learning: An Introduction, MIT Press.