Intelligent Agents

Human Problem Solving and Planning

Ute Schmid

Cognitive Systems, Applied Computer Science, Bamberg University

last change: June 5, 2014

Cognitive Perspective in Problem Solving

- *Early 20th century:* **Behaviorism**
  reinforcement of solutions found by trial and error

- *Middle of 20th century and now:* **Cognitivism**
  trial-and-error cannot explain the systematicity and productivity of thinking (finding new solutions)
  - Most influential researchers:
    Newell and Simon (Human Problem Solving, 1972)
  - *Assumption*:
    search in problem space
  - Weak strategies (uninformed but complete)
    Strong strategies (informed but incomplete)

Research question:
What types of strong strategies are used by human problem solvers?

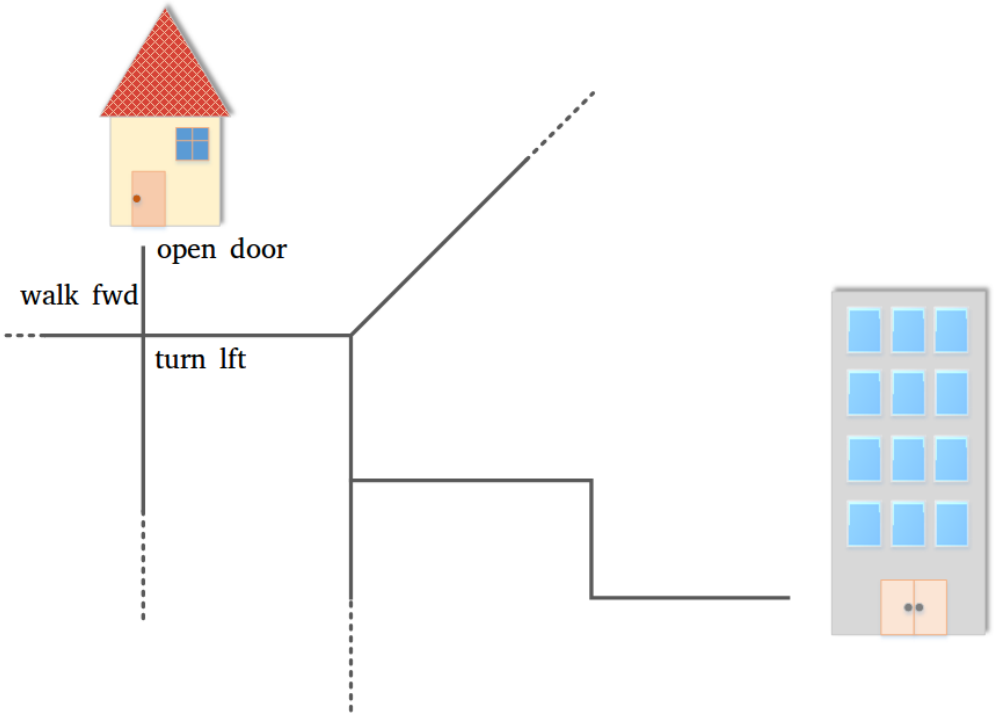
Outline

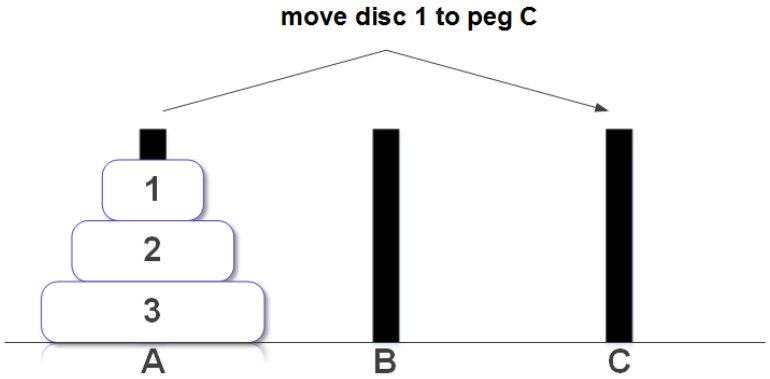
- Aspects of Human Problem Solving
  - Search in a Problem Space
  - Greedy Strategies
  - Means-End-Analysis
  - Cognitive Architectures
  - Finding Good Representations
  - Analogical Problem Solving
  - Learning by Doing
- Hierarchical Planning
  - Problem Decomposition
  - And-Or-Trees
  - HTN-Planning

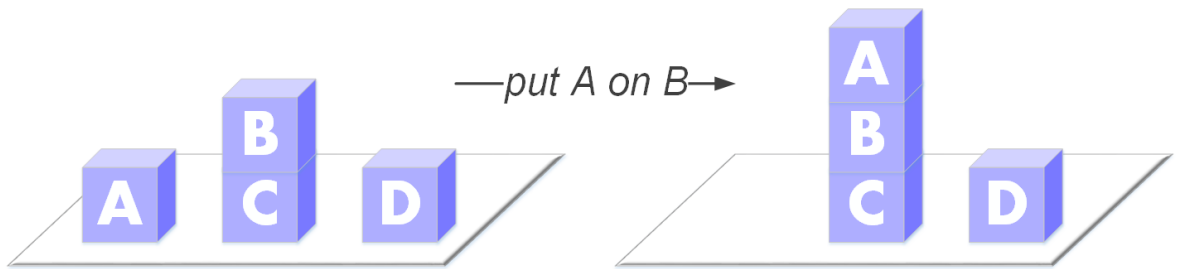
Problem Solving (vs. Planning)

- Cognitive and early AI perspective: Problem solving as search in the problem space
- Focus on search algorithms, not on representation
- Example problems: blocks-world, Tower of Hanoi, Way-Finding
- Focus on well-defined (transformation/interpolation) problems
- Ill-defined vs. well-defined problems (McCarthy , 1956)
  Open vs. closed problems (Minsky, 1965)
- Formal definition of problems corresponds to definition in planning: Given a set of states S which constitute the **problem space**: Initial State ($S_i \subset S$), set of goal **states** ($S_g \subseteq S$), set of operators with applicability conditions $O : S' \rightarrow S, S' \subseteq S$
- Problem solving: Transformation of an initial state into a goal state by applying a (minimal) sequence of operators.
- Problem solving can be performed in a real environment (acting human or robot) or in a simulation (mental process, manipulation of descriptions)

Example Problems







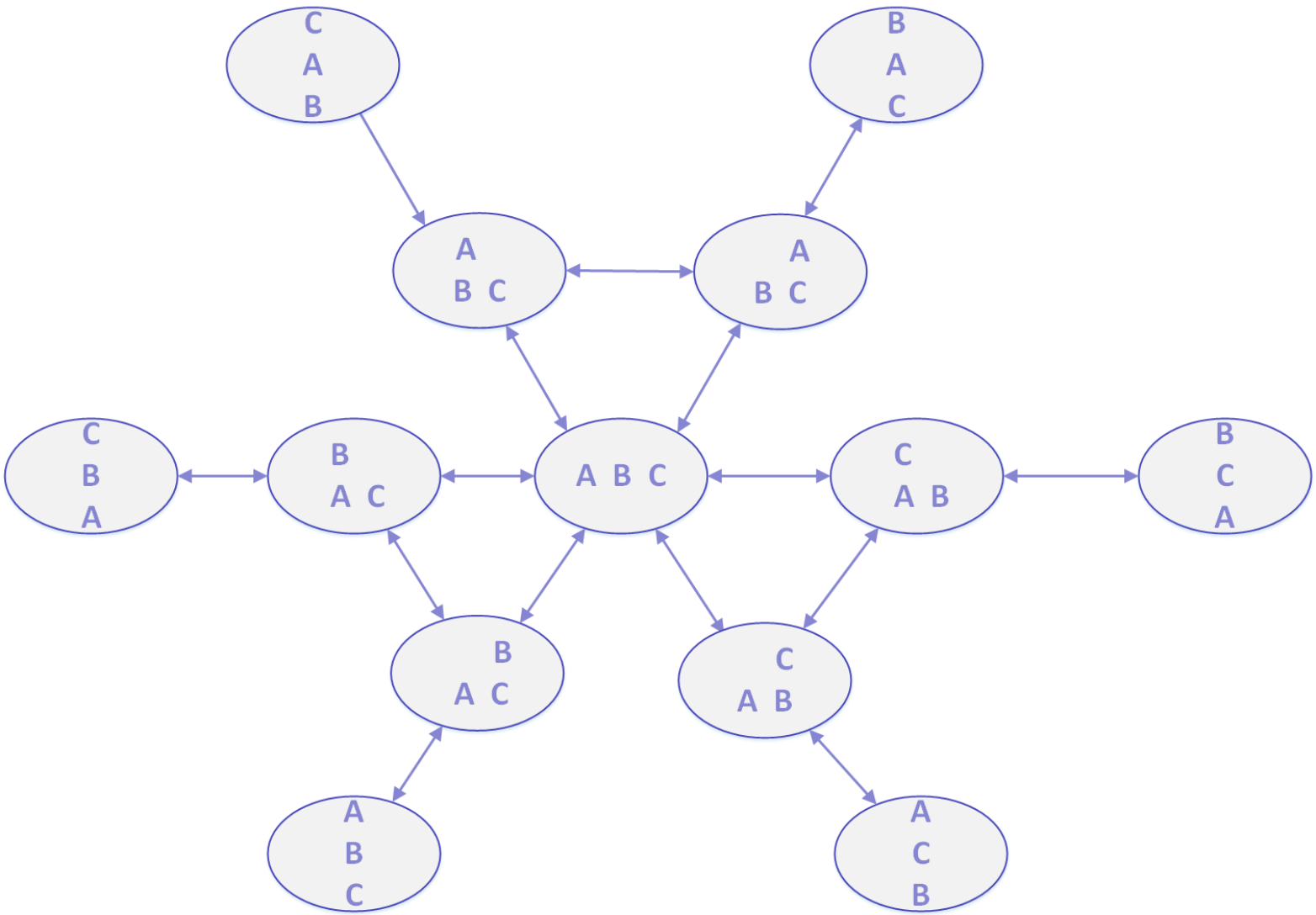
Representation Usually Set-Theoretical

- Example: Way-finding
  - Initial state: *at-home*
  - Goal state: *at-work*
  - Operators: *open door* (if you are in front of door), *walk straight-forward*, *turn left*, ...
- Example: Tower of Hanoi
  - Initial state: *discs 1..3 on peg A*
  - Goal state: *discs 1..3 on peg B*
  - Operators: *put disc on another peg* (if disc is top of peg and if other peg is empty or top of other peg is larger then disc)
- If we can transform an initial state into a goal state by applying **one** operator, we solve a **task** and not a problem.
  (We have the specialized program/the skill to reach our goal directly.)

Problem Space

- Concept of problem space introduced by Newell & Simon, 1972
- Also called state space
- The state space is a graph (S, A) with:
  - S: the set of nodes (all states of a problem)
  - $A \subseteq S \times S$: the set of arcs, with $(s, s') \in A$ iff state s' can be reached from state s by applying an operator $o \in O$.
- **State-space semantics**:
  represents the structure of the problem. Each path in the graph from an initial state to a final state is an admissible solution.
- Note: for computational approaches to problem solving, we do not have the states, but only descriptions of states (which are partial)! We represent only such information which is relevant for the problem (e.g. we are not interested in the material or in the color of blocks).

Problem Space for Blocks-World



Problem Solving as Search

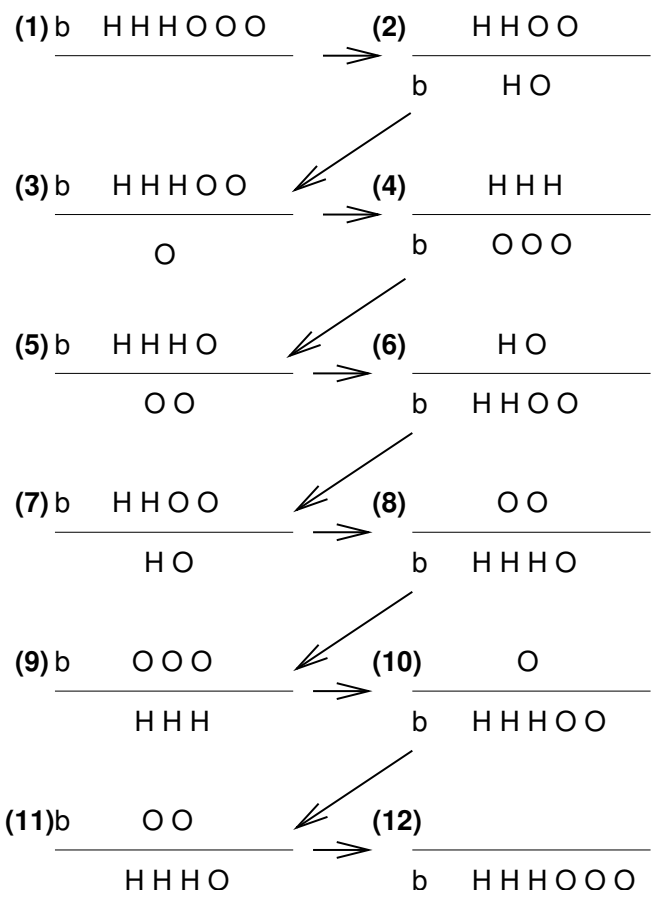
- For interpolation problems we do not know, which sequence of operators is suitable ("admissible") to transform an initial state into a goal state.
- Human problem solvers typically use additional knowledge (besides the legal operators) to solve a problem: choose always that operator which makes the difference between the current state and the desired state as small as possible (heuristics used by GPS, see chapter on human problem solving).
- Algorithmic solution: search

Greedy Strategies

- Humans often use greedy strategies for solving problems ("bounded rationality", Herbert Simon)
- The means-end (MEA) strategy which is the search algorithm used in the GPS (General problem solver) is such a greedy strategy
- Like the original STRIPS planner, MEA uses a linear strategy and therefore is not complete! (Sussman Anomaly)
- Empirical evidence:
  Greeno (1974) for the Hobbits-and-Orcs problem, a simulation with a production system by Schmalhofer & Polson (1986)

Hobbits and Orcs

Subjects have problems with the transformation from state (6) to (7). Here 2 and not only 1 passenger must be transported back to the left river bank. That is, there must be created a situation which is further removed from the goal state than the situation before.



Means-End-Analysis

Transform: Compare current state with goal state

IF the current state fulfills the goal
THEN stop and announce success
ELSE **Reduce** the difference between the current state and the goal.

Reduce: Find operator which reduces the difference between current state and goal

IF there is no such operator
THEN stop and announce failure
ELSE **Apply** the operator to the current state.

Apply: Apply an operator to the current state

IF the operator is applicable to the current state
THEN apply it and **transform** the resulting state into the goal.
ELSE **Reduce** the difference between the current state and the application conditions of the operator

MEA Example: Tower of Hanoi

Transform: initial state (discs 1, 2, 3 on peg A)
to goal state (discs 1, 2, 3 on peg C)
- Reduce: 3 is not on C
- Apply: Move 3 to C
  - Reduce: 3 is not free, because of 2
  - Apply: Remove 2
    - Reduce: 2 is not free,because of 1
    - Apply: Remove 1;
    - 1 can be moved to C
    - 2 is free
  - 2 can be moved to B
  - 3 is free
- Reduce: 3 cannot be moved to C,
- because of 1
- Apply: Remove 1;
- 1 can be moved to B
- 3 can be moved to C

MEA Example cont.

Transform: State (1 and 2 on B, 3 on C) to goal state
    Reduce: 2 is not on C
    Apply: Move 2 to C
        Reduce: 2 is not free because of 1
        Apply: Remove 1;
        1 can be moved to A
        2 is free
    2 can be moved to C
Transform: State (1 is on A, 2 and 3 on C) to goal
    Reduce: 1 is not on C
    Apply: Move 1 to C;
    1 can be moved to C
    1 is on 3
Transform: State (1, 2 and 3 on C) to goal
success

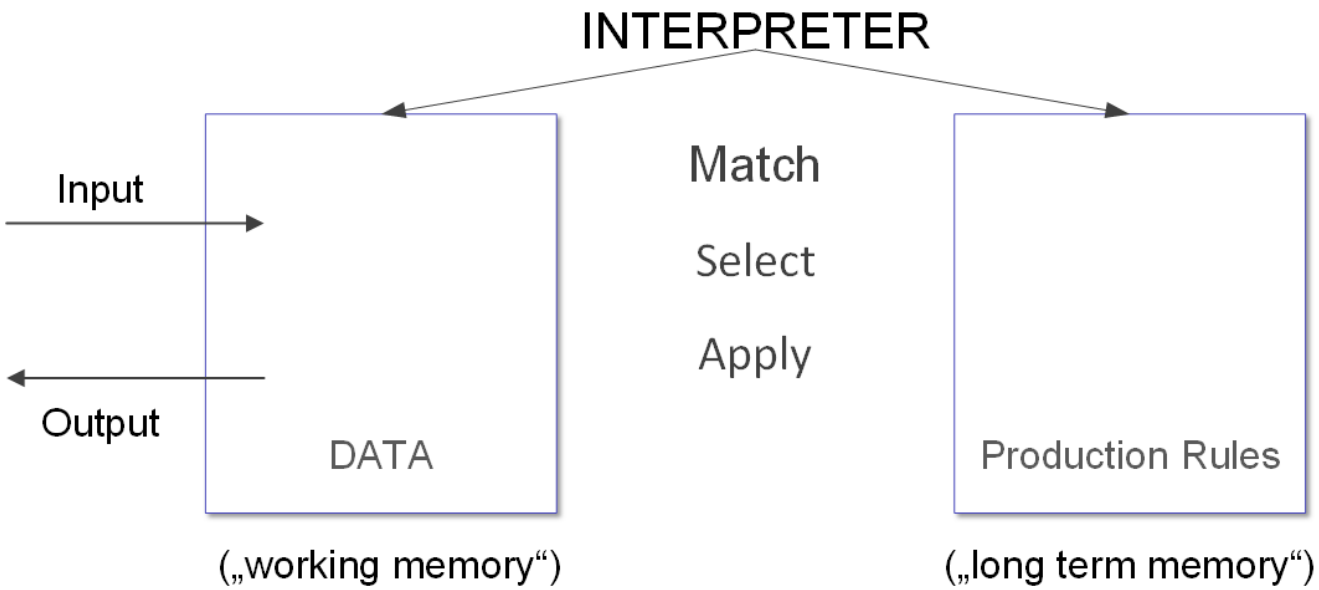
Cognitive Architectures

- Cognitive Architecture: "unified theory of cognition"
  - Explicit definition of basic mechanisms of information processing
  - Assumption that these mechanisms are constant over all domains (problem solving, language understanding, pattern recognition etc.)
  - Basic mechanisms:
    control of interaction with environment, representation of information in memory, strategy to select rules
  - Advantage:
    different models realized in the same architecture get comparable
- Alternative: special purpose cognitive models
  (such as SME for analogical reasoning, see below)

Cognitive Architectures cont.

- Prominent Architectures:
  The ACT-family (J.R. Anderson et al.), Soar (based on GPS)
- ACT and Soar are production systems
- ACT:
  long-term memory is divided in a declarative memory ("know what", activation net) and a procedural memory ("know how")
- Example strategies for selecting rules:
  most specific first, most recently used, priority values (updated in dependence of success)

Production System



Problem Solving Strategy

As long as no final state is reached or there are still reachable, yet unexplored states:

- Collect all operators which can be applied in the current state (Match state with application conditions)
- Select on applicable operator.
  In our example: alphabetic order of the labels of the resulting node.
  In general: give a preference order
- Apply the operator, generating a successor state

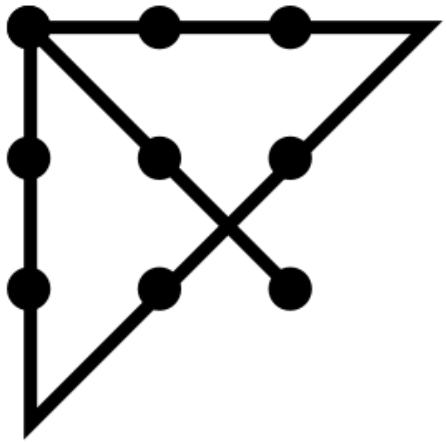
Remark: The Match-Select-Apply Cycle is the core of "production systems" (see chapter human problem solving)

Finding a good representation

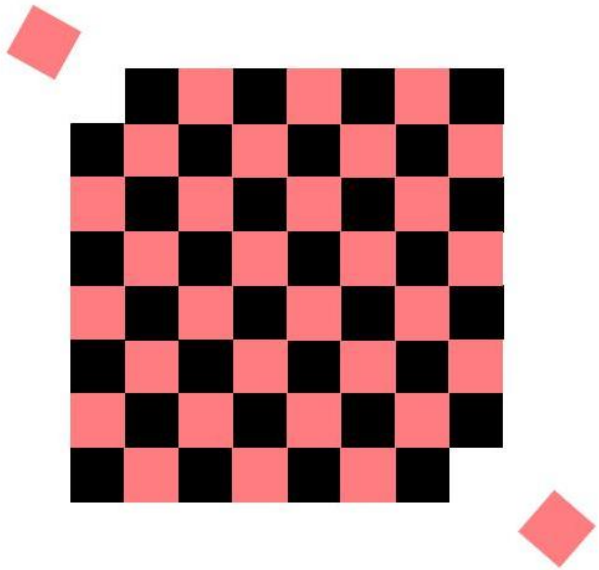
- In human problem solving, there is an interaction between constructing a suitable representation and solving the problem.
- In AI systems, typically the representation needs to be fixed before problem solving (see Kaplan & Simon, 1990).
  Exceptions:
  approaches to solving proportional analogies using re-representation (Copycat, Hofstadter et al. 1995, PAN, O'Hara 1992, Indurkhya 1992)
- Empirical studies: Plötzner & van Lehn, 1997
- Examples: Mutilated checkerboard, nine-dots problem

Examples

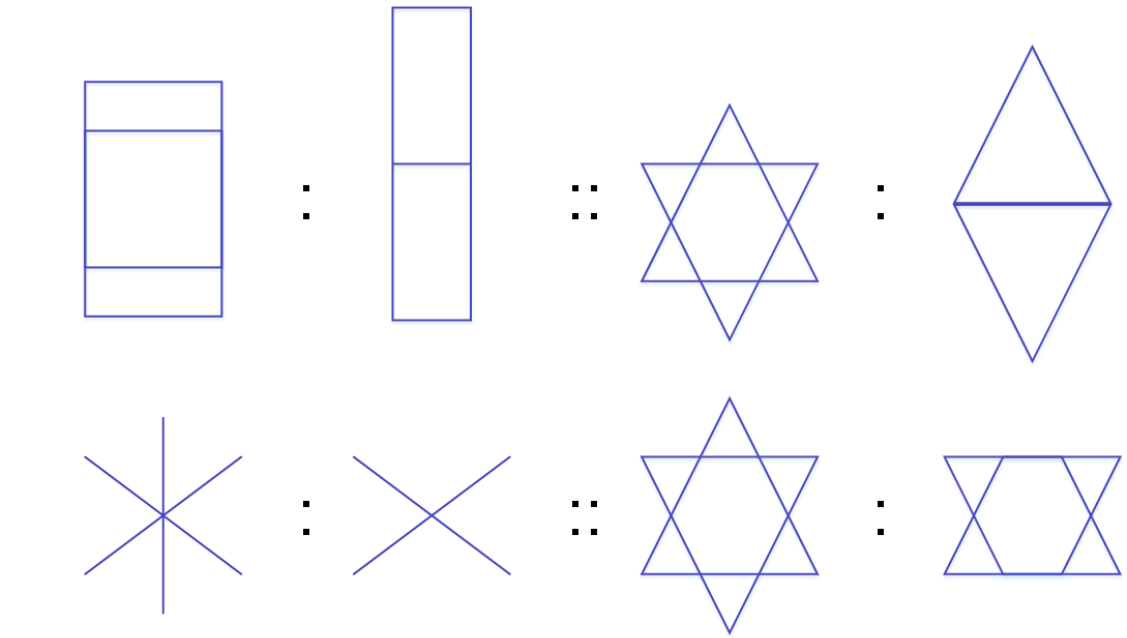
- Nine-dots:

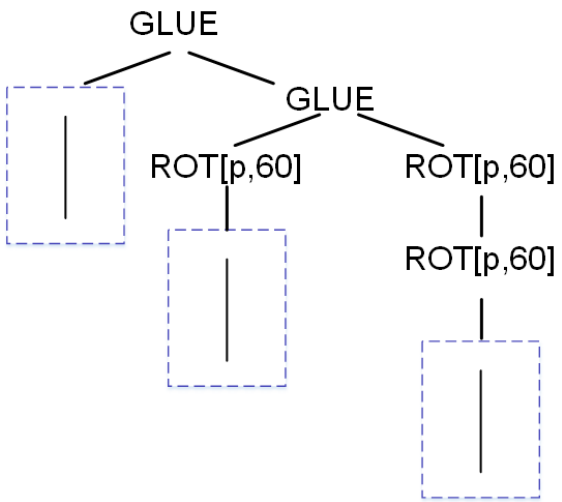


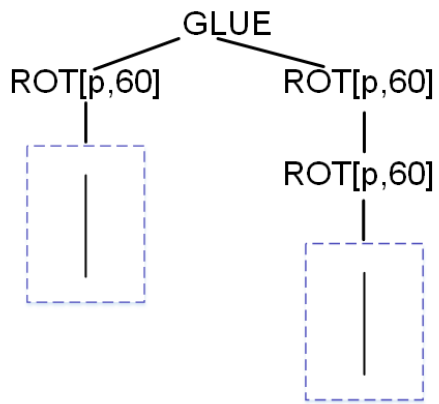
- Mutilated Checkerboard:



Re-Representation







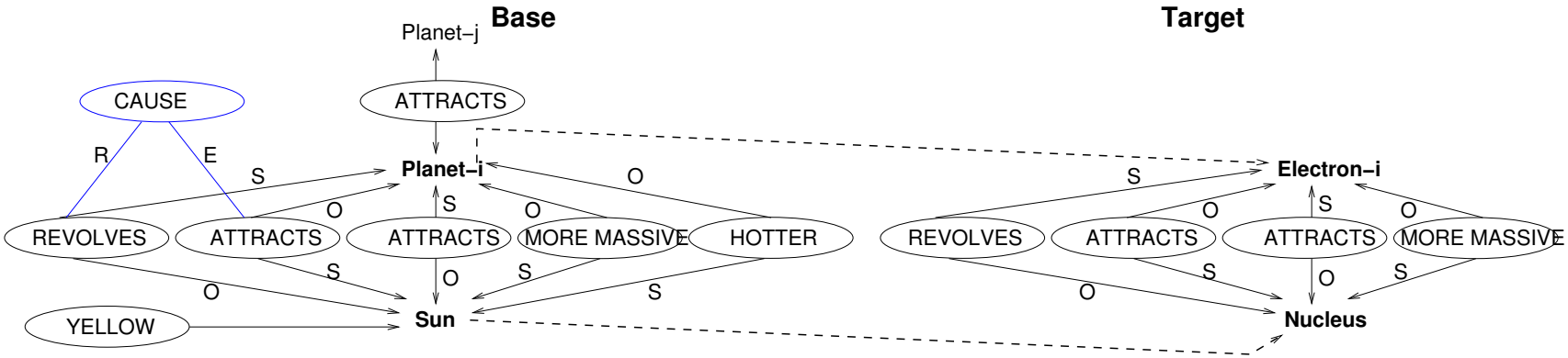
Analogy System PAN (O'Hara)

Context Effects

- Since human problem solving is typically guided by knowledge, search for a solution might be misled by preconceptions
- Gestalt-Theory: functional fixation (Duncker 1945)
  Examples:
  Candle, matches, and box with pushpins; pendulum problem
- A related phenomenon:
  set-effect (Luchins & Luchins, 1950)
  Water jug problems

Analogical Problem Solving

- A problem solving strategy alternative to heuristic search is using analogical reasoning.
  - Retrieve a suitable source problem.
  - Map the entities of the source with the entities of the target problem in a structure preserving way.
- "Carry-over" known parts of the source to target
  (possibly perform necessary adaptations)
- Gentner (1983)
- Cognitive Models:
  SME (Falkenhainer et al. 1989), LISA (Hummel & Holyoak, 1998)
- Empirical investigation of analogical transfer (Schmid et al., 1999)



Learning by Doing

- A problem solving system has no memory. Therefore, it might recalculate solutions which it already had achieved in another problem solving episode.
- The power law of learning (Anzai & Simon, 1979):
  learning curve, speed-up effect
- Humans acquire skills (procedural knowledge) when solving problem
- Cognitive Models, based on production systems:
  ACT (Anderson et al.), SOAR (Newell et al.): Declarative knowledge is "compiled" into rules
- But: these models do not cover strategy learning/control rule learning (see Schmid et al. 2000)

Tower of Hanoi (with IGOR)

Tower of Hanoi can be learned (with IGOR) from three positive examples:

Input to IGOR2

```
eq Hanoi(0, Src, Aux, Dst, S) =
  move(0, Src, Dst, S) .
eq Hanoi(s 0, Src, Aux, Dst, S) =
  move(0, Aux, Dst,
    move(s 0, Src, Dst,
      move(0, Src, Aux, S))) .
eq Hanoi(s s 0, Src, Aux, Dst, S) =
  move(0, Src, Dst,
    move(s 0, Aux, Dst,
      move(0, Aux, Src,
        move(s s 0, Src, Dst,
          move(0, Dst, Aux,
            move(s 0, Src, Aux,
              move(0, Src, Dst, S)))))))) .
```

Induced Tower of Hanoi Rules (3 examples, 0.076 sec)

```
Hanoi(0, Src, Aux, Dst, S) = move(0, Src, Dst, S)
Hanoi(s D, Src, Aux, Dst, S) =
  Hanoi(D, Aux, Src, Dst,
    move(s D, Src, Dst,
      Hanoi(D, Src, Dst, Aux, S)))
```

Figure : Posing the *Tower of Hanoi* problem for IGOR2 and induced recursive rule set

Hierarchical Planning

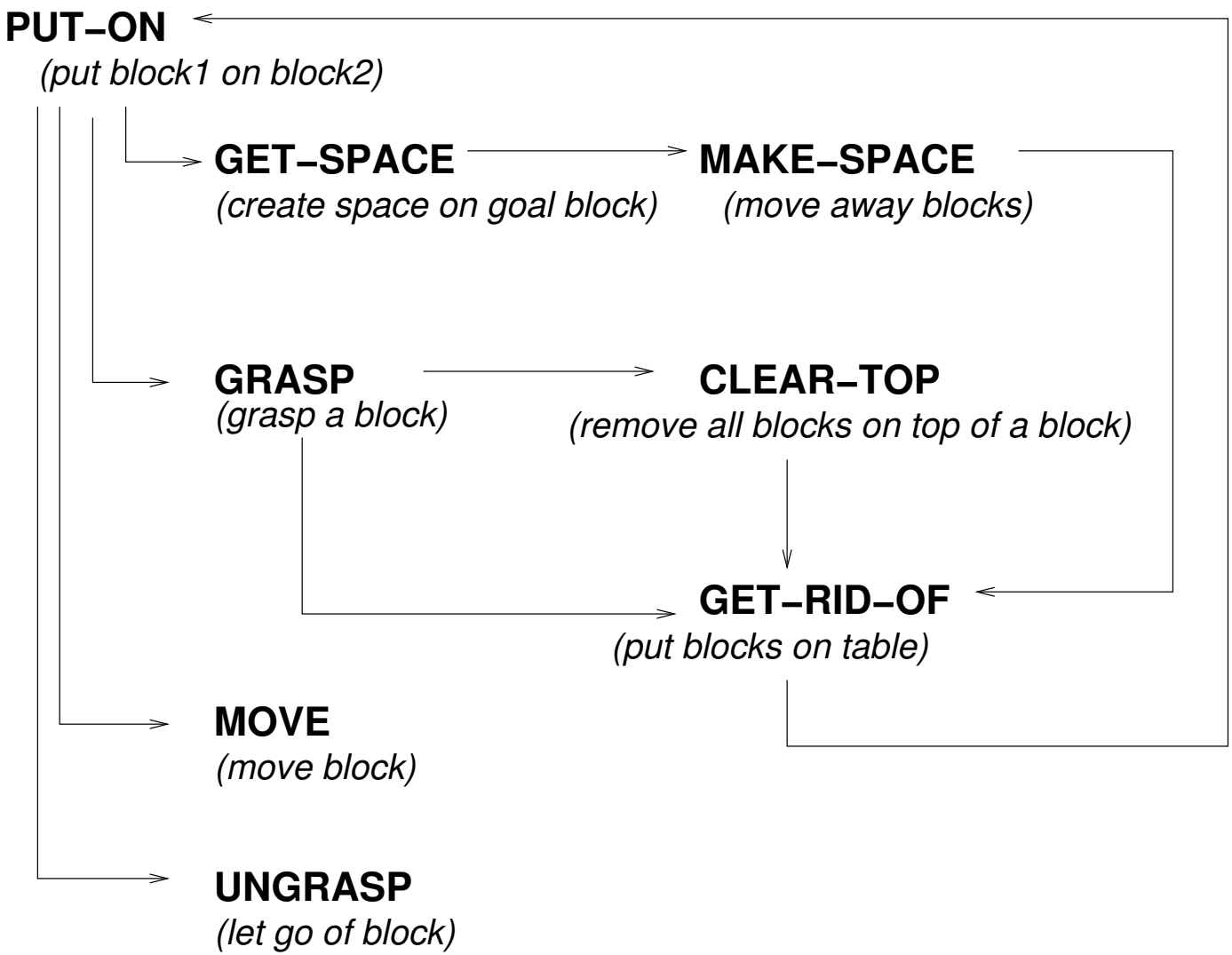
- Humans typically exploit knowledge about a domain when solving a problem
- Often, it is known which sub-goals must be fulfilled to reach a given goal
- This idea is realized in goal-driven production systems (such as ACT-R)
- This idea is also realised in hierarchical planning
- Hierarchical planning is a special case of domain-dependent planning

Problem Decomposition

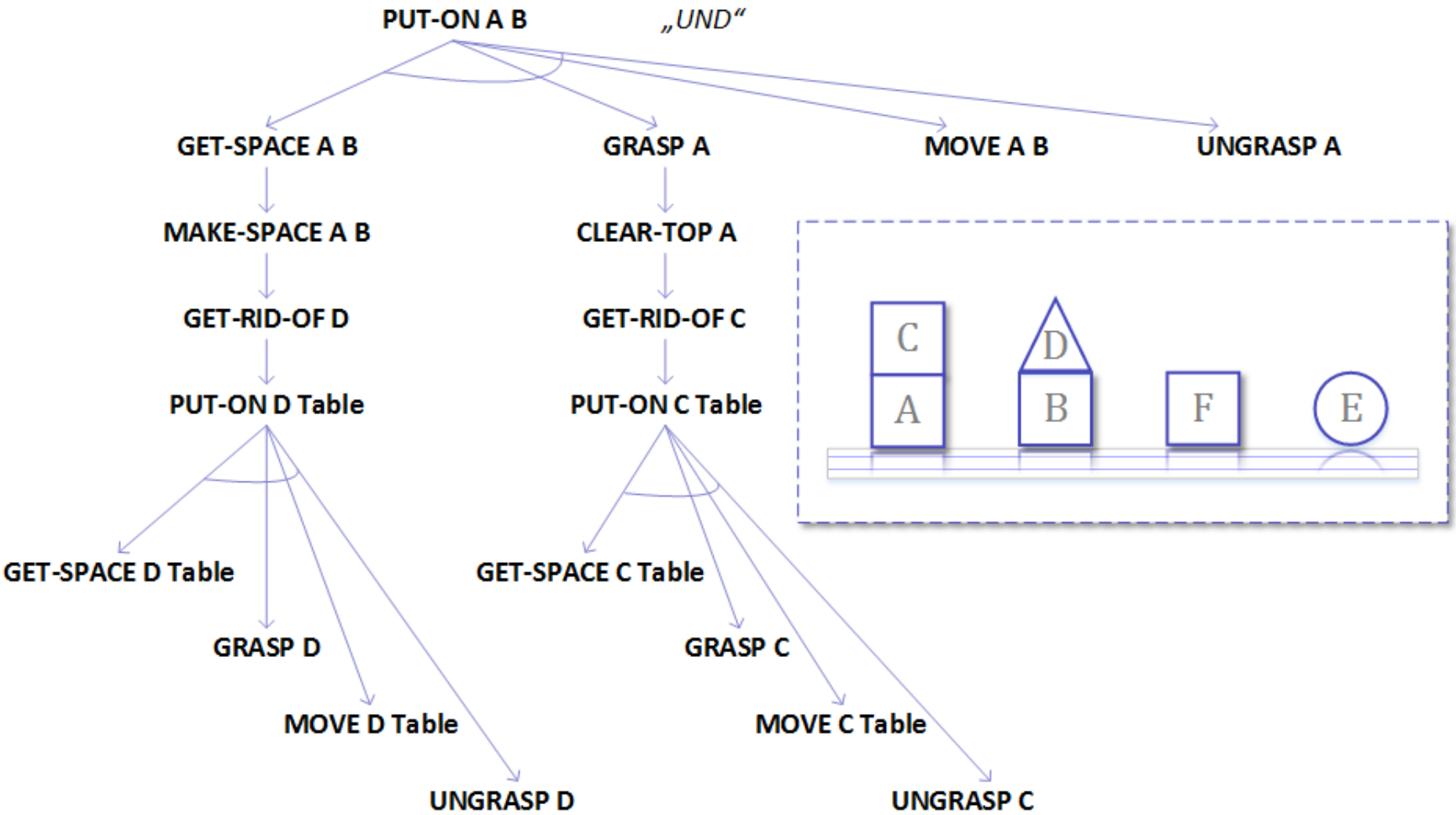
- Besides using heuristics, problem solving can be guided by knowledge about the problem structure.
- Problem decomposition: Dividing a problem in sub-problems
  $\hookrightarrow$ More complex production rules, goal-directed systems
- Advantage: dealing with smaller sub-problems and generating the solution by composition ("divide and conquer")
- Representation: AND-OR Trees
  standard tree: each arc which exits a node represents an alternative ("or");
  extension: specially mark edges which lead to sub-trees which must be all fulfilled for the current node to be fulfilled ("and")
- Special heuristic search algorithm for AND-OR trees: A*

Example: MOVER

(Winston, 1992)



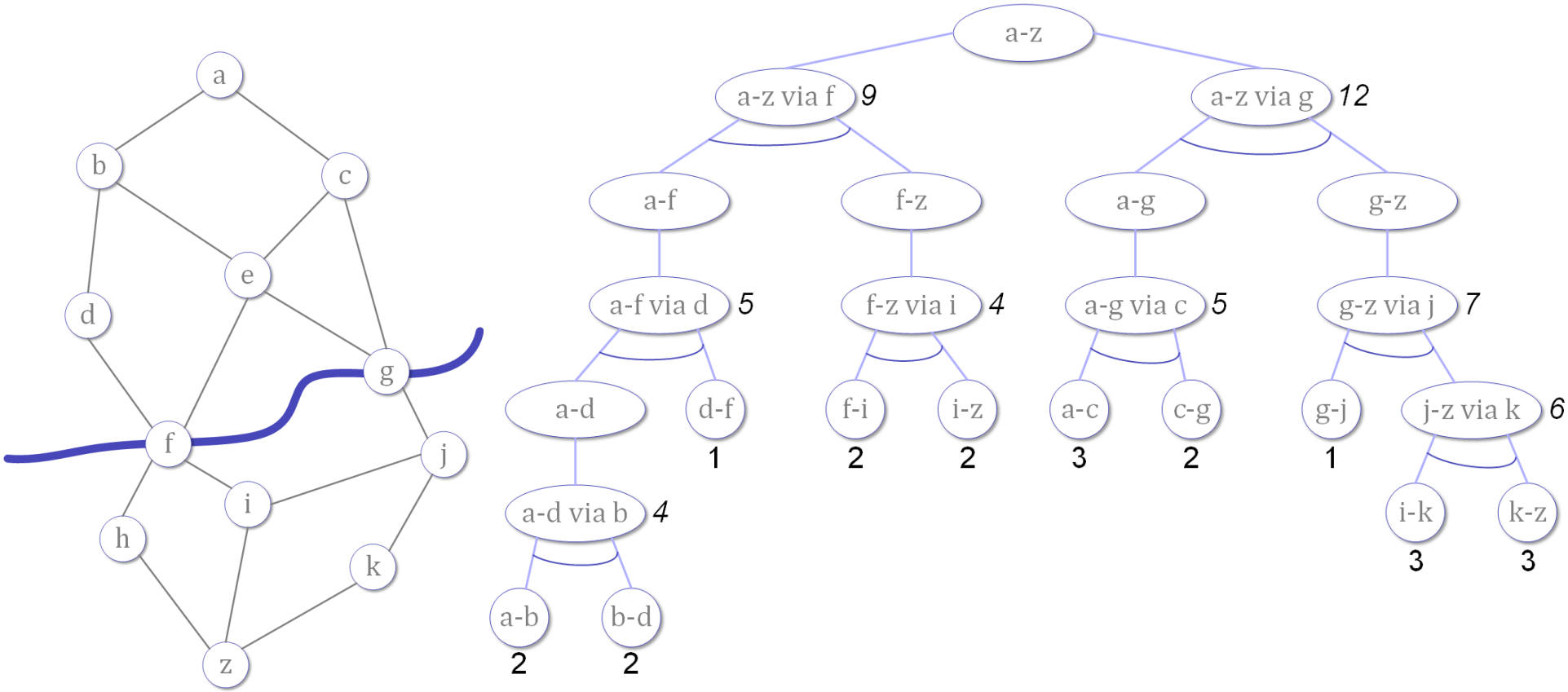
AND-OR Tree for MOVER



Example: Route Finding Problems

- "OR"-node:
  X - F (go from X to F), X - F is primitive, if F can be reached from X in one step (there exists an applicable operation) primitive
  sub-problems are leafs in the tree
- "AND"-node:
  X - Z via Y (go from X to Z via Y) "constraint"
- Problem solving:
  extracting an (optimal) AND-Tree
- Using costs:
  Each leaf is associated with its cost, the costs are propagated upwards in the tree, the AND-tree with the lowest costs is returned
- Algorithm: AO* (Nilsson)
- In planning: hierarchical planning

Example cont.



Two of many possible decompositions.

Summary I

- Cognitive theories of human problem solving are based on the concept of search in a problem space
- There is empirical evidence that humans use greedy strategies (bounded rationalize)
- Means-end-analysis is an early computational model for human problem solving. MEA is similar to the original Strips planner and also incomplete
- Cognitive architectures such as ACT-R rely on a production system architecture. Problem solving is realized by match-select-apply cycles.
- Human problem solving relies on finding good representations.

Summary II

- Humans use analogy as strategy for problem solving
  (search for a similar problem with known solution and transfer it)
- Humans typically learn from problem solving experience
- Humans often use knowledge to solve problems.
- Hierarchical planning is a special case of knowledge dependent planning
- It can be realized by construction of AND-OR-Trees and extraction of an (optimal) solution.