FORMULATING

THE PROBLEM SPACE COMPUTATIONAL MODEL

Allen Newell[1]

Gregg Yost[1]

John E. Laird[2]

Paul S. Rosenbloom[3]

Erik Altmann[1]

[1]School of Computer Science
Carnegie Mellon University, Pittsburgh, PA

[2]Computer Science Department
University of Michigan, Ann Arbor, MI

[3]Information Sciences Institute
University of Southern California, Marina del Rey, CA

Transparencies for CMU Soar meeting, 24 May 1990

Table of Contents

1. Problem spaces. A space of states with a set of operators on states, where a task is formulated as starting at an initial state and searching for a desired state.

2. Logics. A set of clauses that describes the task situation with rules of inference to derive new clauses, where a task is formulated as proving a given theorem from a set of axioms.

3. Constraint satisfaction. A set of variables of given (often discrete) ranges subject to a set of constraints of fixed types, where a task is formulated as finding the values of the variables that satisfy the constraints and (often) optimize various functions of the variables.

4. Schemas. A hierarchical attribute-value structure with values determined by inheritance hierarchies, defaults, constraints, and attached procedures, where a task is formulated as filling out the slots of a given schema.

5. Programs. A programming language, where a task is formulated as synthesizing a program to meet a specification on the inputs, outputs and behavior. The specification is given in some formal language.

6. Plans. A (large) data base of highly abstract variabilized procedures containing subgoals to be obtained by other plans, where a task is formulated as attaining a given partially instantiated plan.

7. The big switch. A (large) data base of programs with a big index on task-relevant features, where a task is formulated by sorting it through the index and applying whatever program is obtained.

Figure 1: Candidate frameworks for formulating tasks.

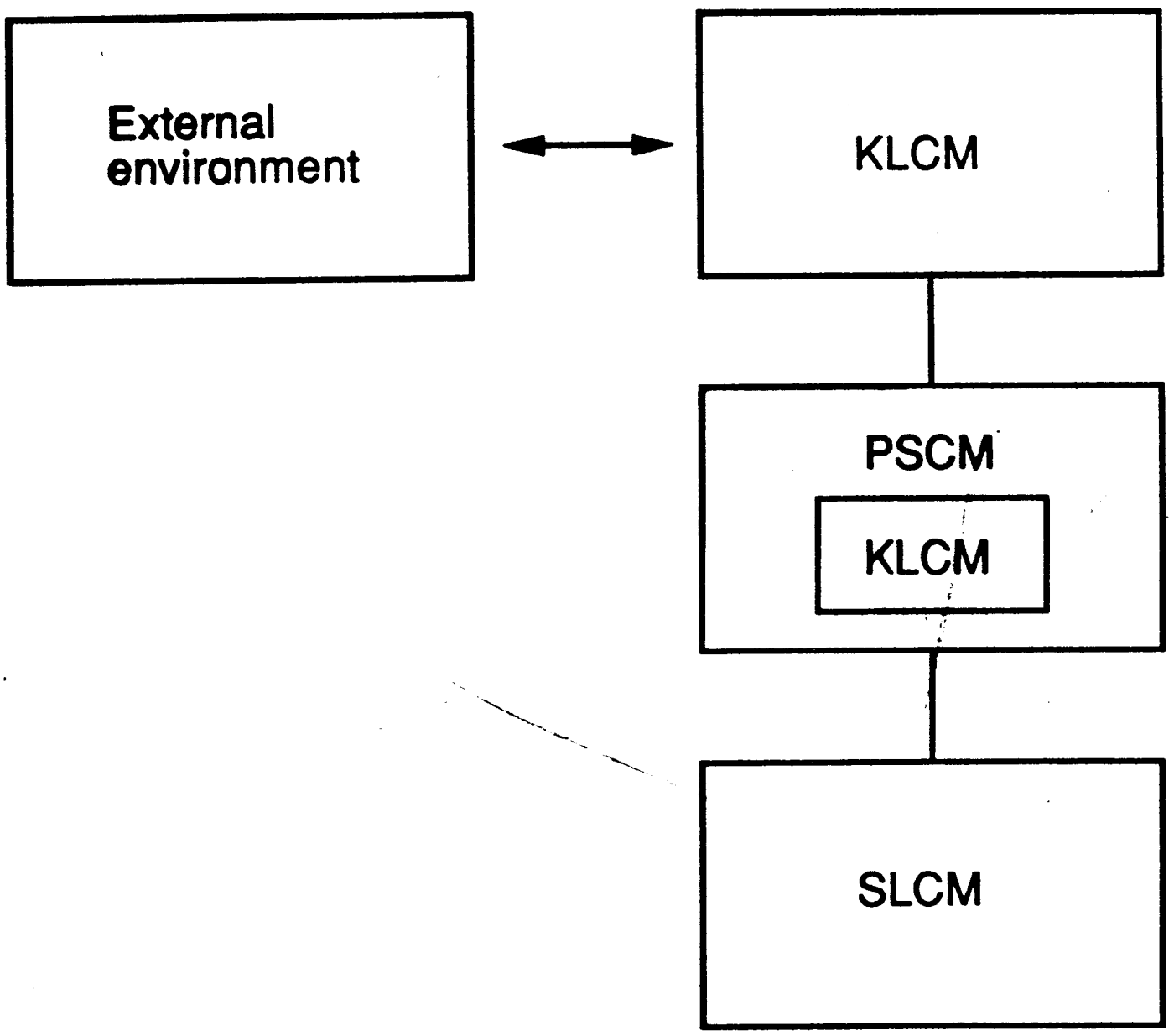


Figure 2: Computational models involved in defining problem spaces.

E: An external environment
Exists in some state at each time t
The agent is an object in the environment

The agent

A: A set of actions
Can be evoked at time t
May or may not be applicable, depending on state of E
Affect state at next time t+1

P: A set of perceptual devices
Adds knowledge of E to K at time t
This is the only source of new knowledge

G: A goal

K: A body of knowledge
About the environment, goal, actions, perceptual devices

Principles of Operation

Rationality: Take actions that know lead to attaining goal
Take A_t such that K_t and K(G) imply will-attain(A_t, G)

Permanence: Knowledge once acquired is available ever after
P_t:$K_t \rightarrow K_{t+1}$, where K_{t+1} includes K_t

Figure 3: The definition of a knowledge-level (KL) system

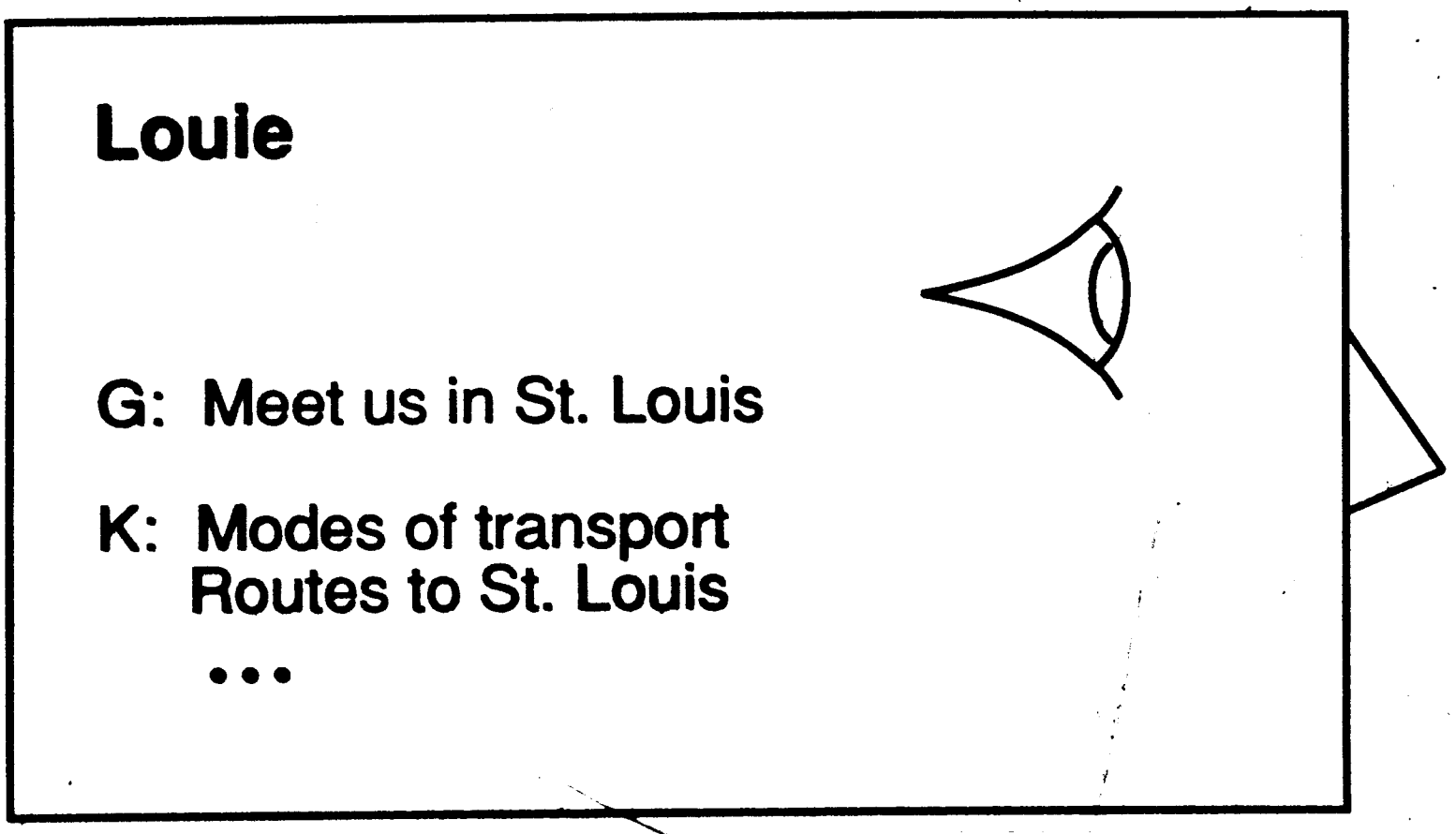


Figure 4: Example KL system: an intelligent system.

E: Personal computer

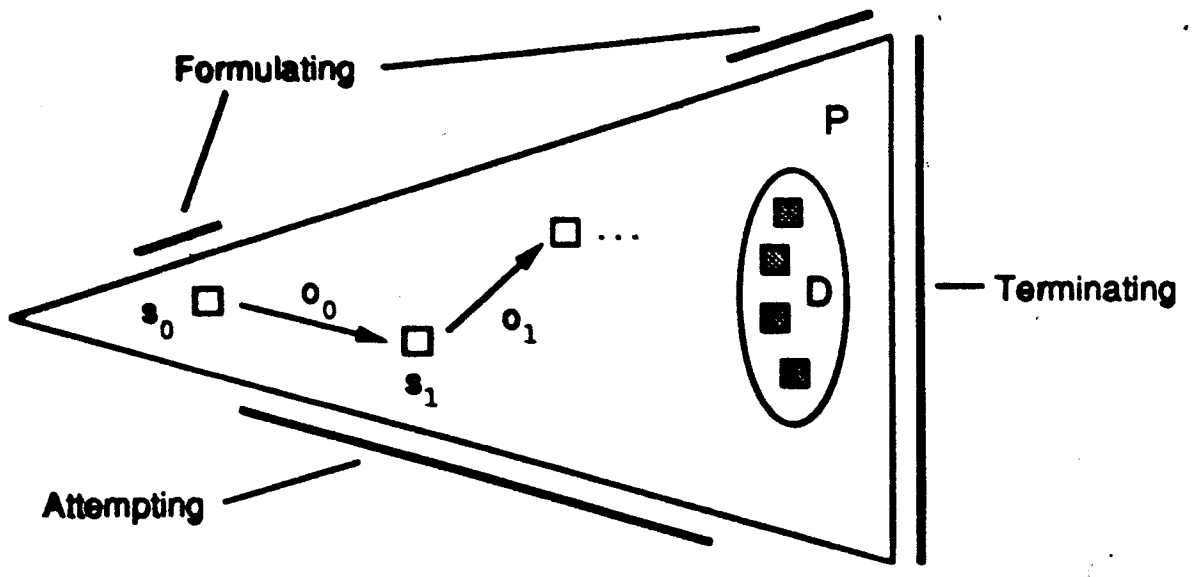
G: Prompt for a query, then respond to it

K: A database relating townships, electoral districts, zip codes
The ability to retrieve on each key

P: Strings that represent queries
Format: "I ; Q"
Where I is an index, of the form:
"D = d" or "T = t" or "Z = z"
Where:
d is a district number
t is a township
z is a zip code
and Q is a query, of the form:
"D = ?" or "T = ?" or "Z = ?"
Examples:
Q1: "D = d; Z = ?"
(Given district number, what are all zip codes in it?)
Q2: "Z = z; D = ?"
(Given zip code, what districts is it in?)

A: Responses sent to the printer
Examples (from above):
R1: List of zip codes
R2: List of districts

Figure 5: Example KL system: a design specification.



A problem space P is **defined** by:

- S A set of states
- O A set of operators on S
  Each o in O is applicable to a subset S_o of S
  Each o realizes a function: $o:S_o \rightarrow S$
  Applying o to a state in S_o produces a new state in S

Behavior occurs in problem space P by consecutive steps:

The step at i starts with the system at current state s_i

Each step has two phases:

Selecting: The current operator o_i is selected from O

Applying: The operator o_i is applied to s_i

If o_i is applicable, the current state at i+1 is $o_i(s_i)$

If o_i is not applicable, still remain at s_i

A task is **formulated** using a problem space by:

Determining-space: A problem space P is adopted

Goal-setting: A set of desired states D is adopted

Initializing: The state s_0 becomes the current state

A formulated task is **attempted** by:

Solving: Steps are taken in the space

Independence principle: Any operator that can be applied, may be applied

Terminating: The task attempt is terminated

If the current state s_i is in D

If s_i cannot be brought to be in D

Figure 6: Definition of a problem space

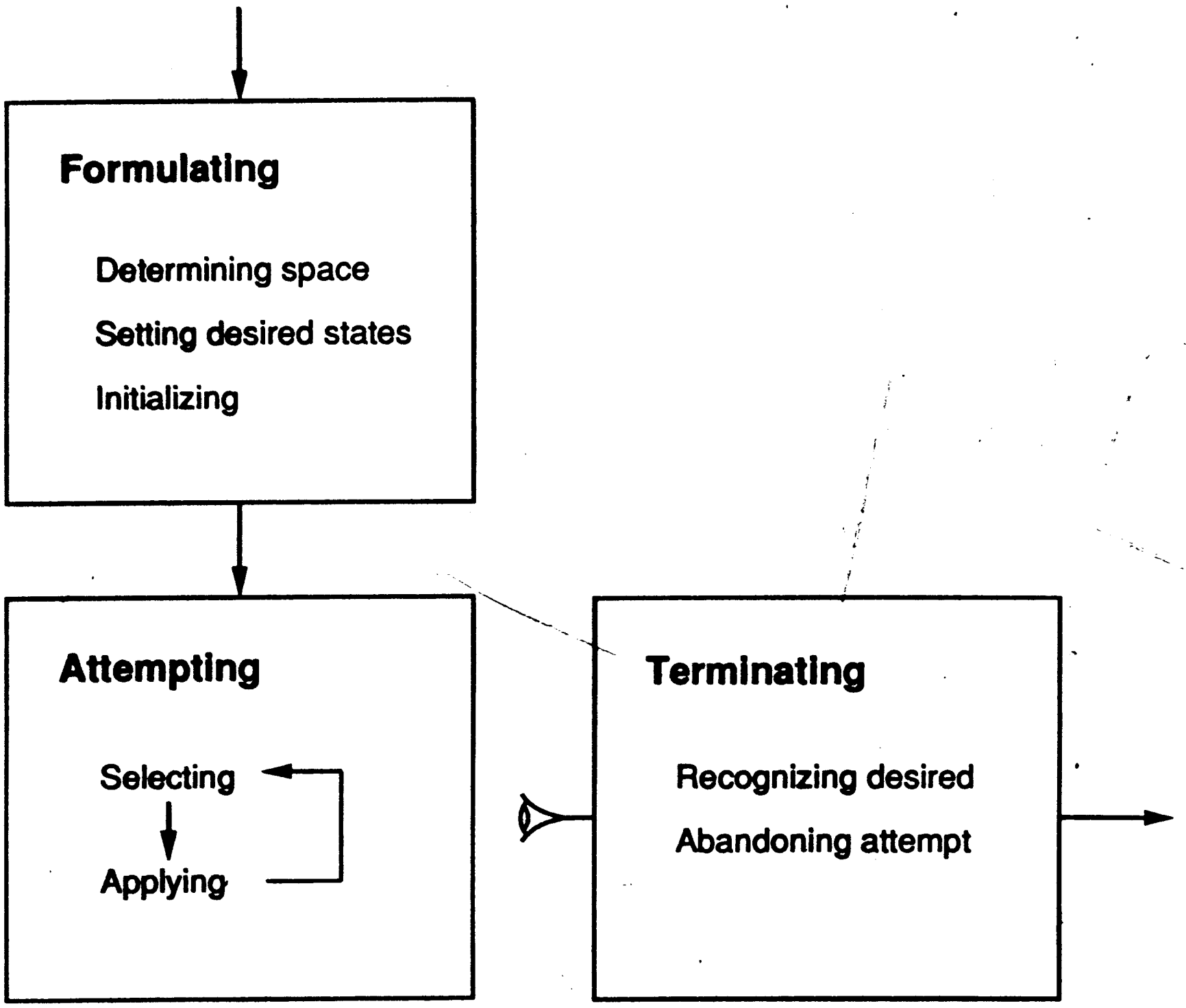


Figure 7: Problem space processes.

Environment	The inner computational context of an agent.
Formulate-task	
Goal	To formulate a task by using a problem space
Actions	Produce P (the problem space), D, s_0
Perceptions	Of the current context, the impulse situation
Knowledge	About problem spaces, the current context
Select-operator	
Goal	To select o_i to reach a state in D
Actions	Produce o_i
Perceptions	Of s_i
Knowledge	About P, s_0 and D
Apply-operator	
Goal	To apply o_i to s_i
Actions	Produce $o_i(s_i)$
Perceptions	Of o_i and s_i
Knowledge	About the conditions of applicability of o_i, the function $o:S \rightarrow S$
Terminate-task	
Goal	To stop attempt if s_i in D or if cannot succeed
Actions	Stop processing
Perceptions	Of P, D, o_i, and $o_i(s_i)$ for the current i
Knowledge	About the current context

Figure 8: Knowledge-level specification of problem-space components.

System Specification

Name: Job-shop-scheduler
State:
Time slots: 1, 2, 3
Objects: A, B
Schedule: (<object>, <machine>, <timeslot>), ...
S_0:
A: rough, triangular, no holes
B: spherical, no holes, surface undetermined
D:
A: cylindrical, polished, size 2 hole
B: cylindrical, polished, no holes

Operators:
Lathe (what when)
Preconditions: none
Effects: cylindrical, rough
Punch (what when size)
Preconditions: unheated or rectangular
Effects: rough, hole
Polish (what when)
Preconditions: unheated or rectangular
Effects: smooth
Roll (what when)
Preconditions: none
Effects: cylindrical, hot, surface undetermined

Operator selection:
Rule out all unhelpful operators
Those that do not achieve a desired characteristic
Punch before rolling
Rolling heats the object so it cannot be punched
Punch or lathe before polishing
Punch and lathe leave an object rough
Try earlier time slots first
Choose randomly among undifferentiated candidates

Behavior on the task

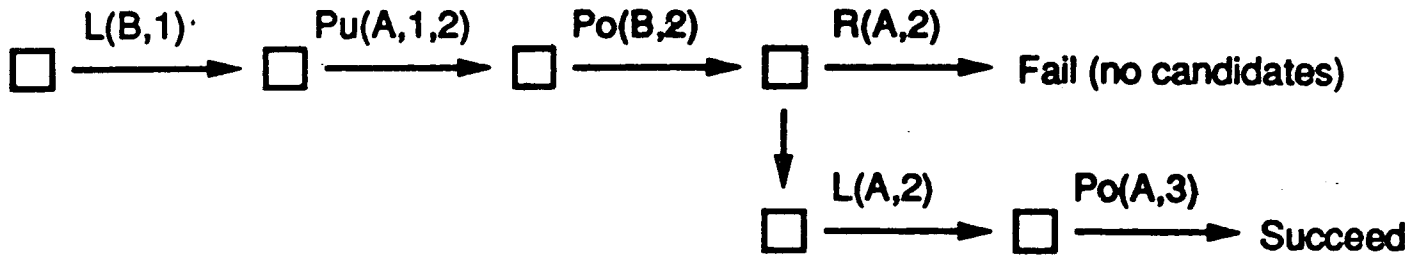


Figure 9: Example of a problem-space system.

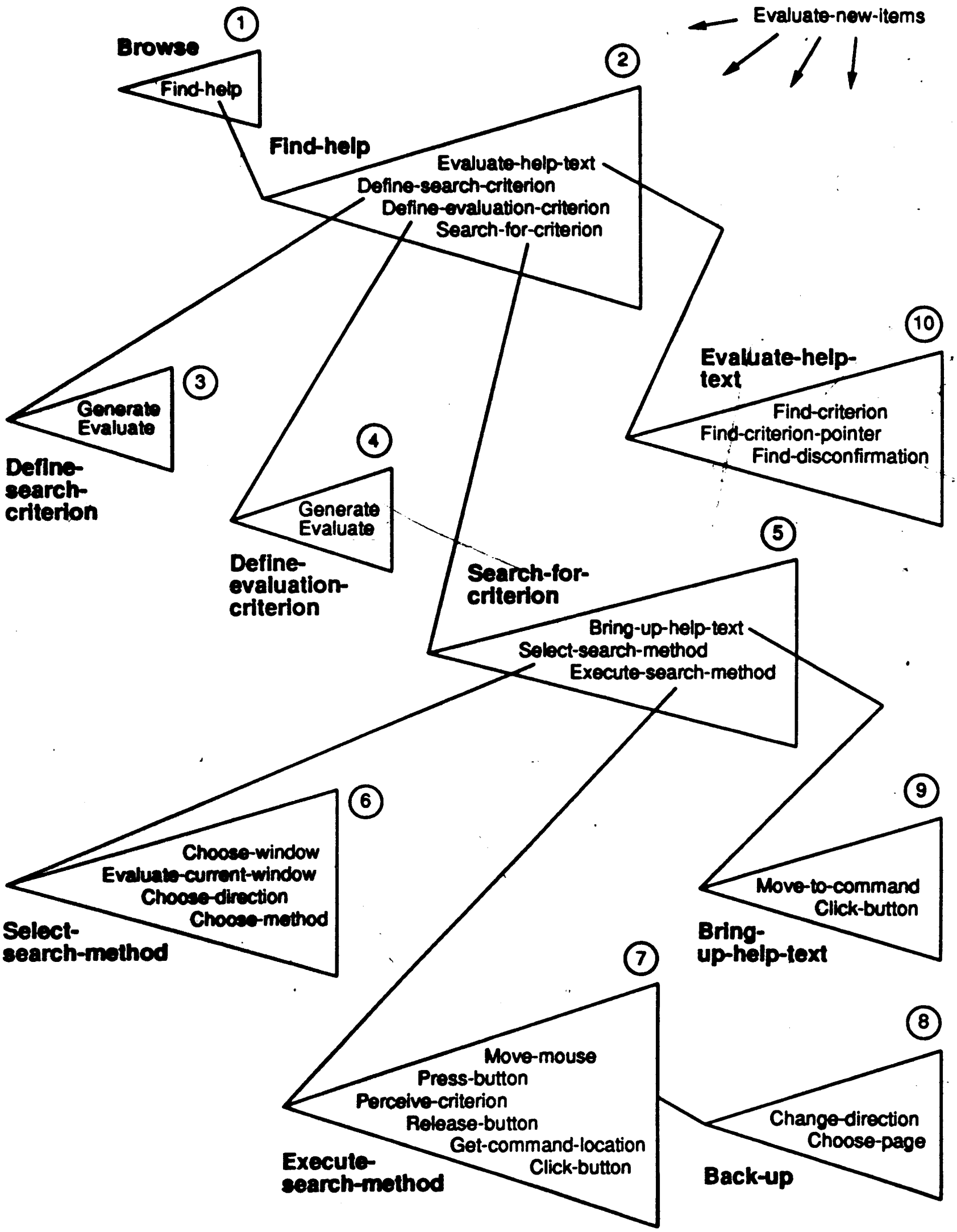


Figure 12: Problem-space trace of Browser-Soar

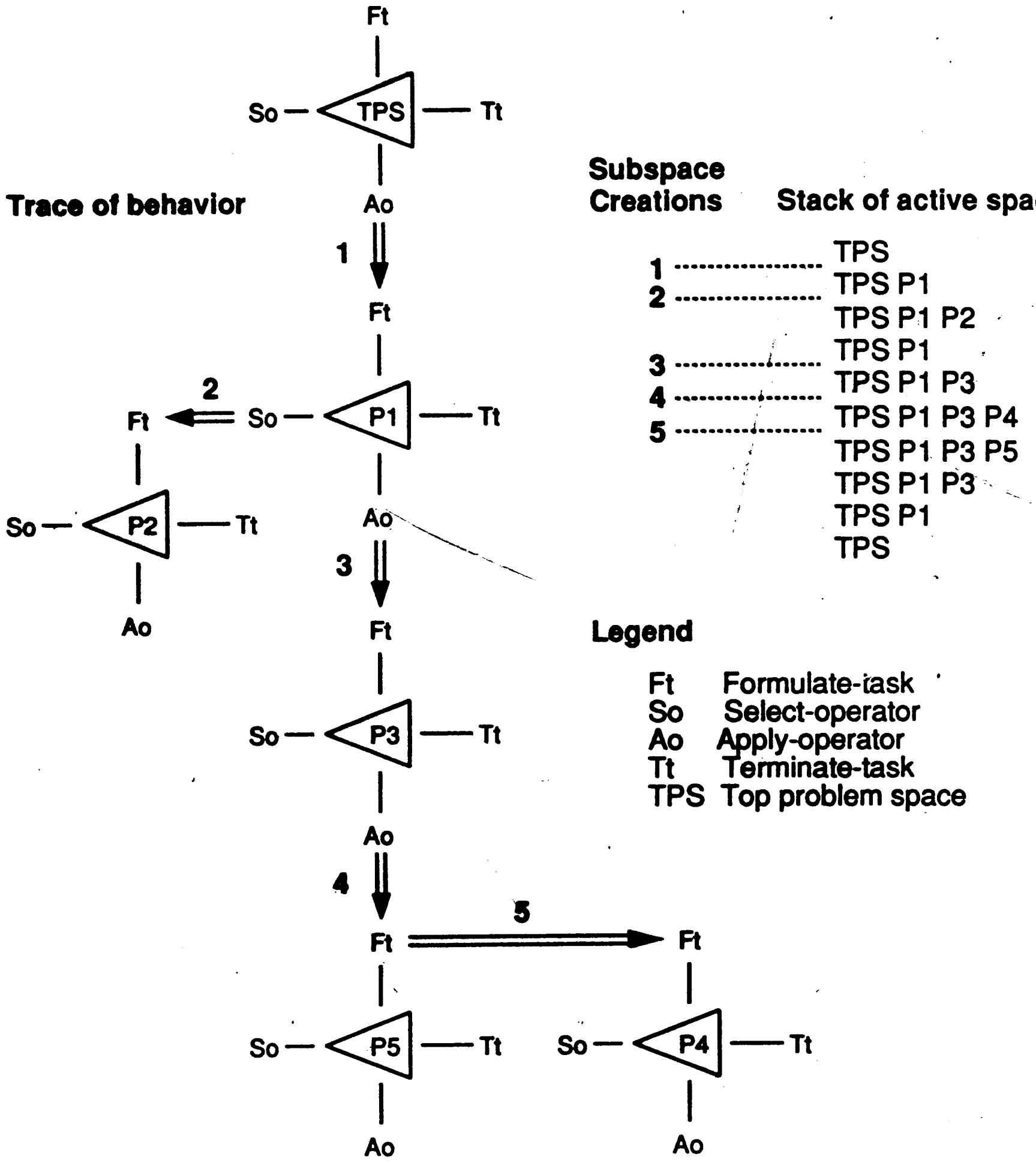


Figure 13: Operational details of PSCM operation

Instructions:

1. Read four premises.
2. Then read a statement.
3. If the statement is true say "true".
4. Then stop.

The specific task:

5. A plate is left of a knife.
6. A fork is left of the plate.
7. A jug is above the knife.
8. The fork is below a cup.
9. The cup is left of the jug.

The system should answer:

10. True.

Figure 14: Simple instruction-taking (by Soar)