International Journal of Computer and Information Sciences, Vol. 10, No. 2, 1981

Sense-Controlled Flexible Robot Behavior

Harri Jäppinen[1]

Received October 1980; revised March 1981

The creation of physical behavior by computational means has been approached differently by industrial and artificial intelligence robotics. Industrial robotics, considering fast response of a robot its most important characteristic, has equipped the robot with predefined, specific behavioral trajectories resulting in fast but inflexible behavior. Artificial intelligence robotics, claiming flexibility as the paramount robot feature, has employed inferred behavior whereby the robot itself determines behavioral patterns for tasks based on the robot's general knowledge about a task domain. Response is now flexible, but the response time is commonly badly degraded. This work defines an action propensity "skill" which generates flexible *and* fast behavior. Flexibility is achieved by attaching "perceptions" in skills to guide behavior; fast response results from the direct activation of skills. The acquisition and generalization of skills happens under the supervision of a human teacher in an advice-taking mode into which the robot shifts from the execution mode after recognizing lacking competence for a given task. This paper defines such skills, describes an implemented skilled robot system, and discusses some simulation results.

KEY WORDS: Skill acquisition; intelligent behavior; procedural knowledge; robotic planning; supervised learning.

1. INTRODUCTION

The creation of nontrivial physical behavior by artificial means, an old problem for the imaginative engineering mind, received new impetus with the invention of the computer. Different ways of weighting two conflicting features, efficiency and flexibility, have bifurcated research into industrial and artificial intelligence (AI) robotics. Research of the former has been satisfied with robots that can repeatedly perform predefined specific trajectories, while the *sine qua non* of AI robotics has been a system's ability to

[1]Helsinki University of Technology, Digital Systems Laboratory, Otakaari 5 A, SF-02150 Espoo 15, Finland.

create necessary trajectories *itself* using general knowledge about a given task domain.

Both approaches have their strengths and weaknesses. While the industrial robot's response is fast, it is rigid; and while the response capability of an AI robot is general with respect to a given task domain, it is so at the expense of badly degraded response time caused by planning before acting. Due to usually exponentially increasing planning time, so far designed AI robots have been confined to strongly restricted task domains.

This work aims at a more balanced system where both generality *and* speed of response are taken into account. This is achieved by employing an evolving set of *general* action propensities. These action propensities, henceforth called "skills," are generated and transmitted to the robot in advice-taking mode by a human teacher.

The robot recognizes when it lacks a necessary skill for a (sub)task and then asks its master to teach a new skill. Due to generality of skills and their hierarchical and modular structure, although such a robot will initially frequently request help, its need for help will gradually decrease while its competence increases.

Undoubtedly the important feature, the one by which such an approach stands or falls, is the degree of generality skills capture. Intelligence implies flexibility, and unless each skill can take care of a set of different situations of a given task domain, skills represent just a collection of generators of ad hoc behavioral patterns.

The solution of flexibility and generality of skills in this work is to abandon the prevailing view of both industrical and AI robotics that generators of trajectories (programs or plans) refer to specific cartesian or joint coordinate points. Skills refer, instead, *indirectly* to state of the surrounding worls through the act of "perceiving." Rather than having control information attached as absolute values in the bodies of skills, control information for skills is mediated by perceptual expert procedures whose calls are attached to the bodies of skills. Perceptions yield generality, since a proximal perception such as touch is invariant over mutual displacements of a perceiver and a perceived object, and a distal perception such as vision is in addition invariant over distancies between the perceiver and the perceived object.

Such skills are general when contrasted with programs used in industrial robotics in the same way as computer programs using variables as placeholders of concrete values are more general than calculators. Calls of perceptual experts, too, are place-holders of possible perceivable states of the world. The state of the surroundings at the moment of behaving provides concrete "values," an interpretation, for skills.

The purpose of this work is to define such a skill and implement and simulate it as an interactive robot-man system. The robot's competence is then almost entirely dispersed in an evolving set of perception-controlled skills; it possesses only a minimal declarative world model for orienting purposes. The robot's response for tasks within the robot's competence is fast. Since the robot "perceives" its environment, it can also interact in dynamic worlds.

In brief, the salient ideas of this work are (i) to use perceptions to directly guide behavior (based on actual states of the surrounding world) and (ii) to form a human-robot advice-taking system in which the robot's competence is gradually increased in the most efficient form (as procedural skills).

2. THE CONVENTIONAL AI ROBOT IN BRIEF

The conventional AI robotics approaches behavior generation from the viewpoint of inferrence. A paradigmatic AI robot system represents a triplet (PS, WM, M) of a *problem solver* PS, a symbolic representation of a task domain WM (*world model*), and a *monitor* M. The world model usually describes both the current state of the task domain and the ways the robot's operators change states.

A task is commonly expressed as a utopian state of the task domain. PS, then, reflects on WM, testing alternative sequences of operators, in an attempt to find a *plan* as a sequence of operators whose execution will transform the initial state of the task domain into the utopian state. When a successful plan is found, it is passed to M which interprets the plan, executes its operators, and continuously checks the correctness of the ongoing execution.

This approach carries some fundamental problems. For one, how to avoid the combinatorial explosion in search of operator sequences while expanding a task domain. This concern is so paramount that the planning speed of a robot system is commonly used to judge the quality of proposed systems. For example, Sacerdoti(11) reports how by imposing a semantic dimension "criticality" upon operators of STRIPS he was able to drasticly reduce the planning speed in his ABSTRIPS system. But still it took over 6 minutes for ABSTRIPS to find a plan of 11 operators. (For the same plane STRIPS used over 20 min.)

Another major problem of inferred behavior is how to guarantee the reliability of WM on which the control information of generated plans is based, and what precautions to implement in *M* to handle inevitable occasional mismatch between the model and reality.(2,5,10) This problem results from the bifurcation of the behavioral world into external or actual world states, and their internal or believed representations in the world model. Since plans thoroughly rely on specific believed states of the world, rather than actual states, surprises cannot be totally avoided.

In inferred behavior architectural problems are also formidable. Building of a complete discrete world model poses hard problems even for restricted task domains. Fahlman,(1) for instance, reports how tedious it is to arrest in a WM the effect of gravitation to rathers simple block constructs. How might a robot infer how to tie shoelaces!

Tangwongsan and Fu(13) introduce an interesting deviation from the paradigmatic robot system, in fact a system which in part resembles our system. A human teacher provides example plans for tasks, and the robot attempts to utilize them in analogous tasks. Chaining of operators in the plan formation phase is thus avoided and "planning" is greatly speeded up, as the authors demonstrate. The remaining main difference with our work is that while their system utilizes plans, operator sequences referring to states of a world model, our system, to repeat, utilizes skills which are controlled by actual world states. Section 8 compares the supervised learning strategies of both systems.

Contrast then inferred behavior with human behavior. Everyday human behavior does not derive from thorough use of inference. We make plans, to be sure, but only on a gross level while leaving control details in the care of sense-controlled skills. This behavioral strategy is particularly good in that it allows us to cope with the boundless world and drastically reduce surprises. An extensive use of skills also results in efficient behavior. In the formation of these skills repetition, imitation, trial-and-error, and other noninferential means play major roles. Such natural skills this work attempts to imitate by computational means.

3. TASK EXPRESSIONS IN A SKILL SYSTEM

In a skill system a task is expressed as a list of *requests*. Each request indicates a subtask for a robot, and the robot uses the request in associatively fetching the corresponding skill, a behavioral expert for solving the subtask.

Assume a robot has been built to work as a servant in a house; it should manipulate objects in the house as requested. In fact, this task domain was used in the simulation. Below appears an example task expression requesting the robot to bring a chair from room 204 into the robot's initial room and to leave it by a table. "*" indicates the robot's response to requests.

```
(! GO INTO R204)
*OK. AND THEN?
(! TAKE A CHAIR)
*OK. AND THEN?
(! COME BACK)
*OK. AND THEN?
(! GO TO A TABLE)
*OK. AND THEN?
(! LEAVE THE CHAIR BY THE TABLE)
*OK. AND THEN?
NIL
```

The robot has "understood" the master's requests, enclosed in parentheses, in the sense that it has found corresponding skills. A failure in the fetching of a skill would initiate the teaching process. The master's NIL signals the end of the task expression. Should the robot be frequently requested to perform such tasks, the skill sequence above could be stored in the general form for bringing a named object from a named target room into the robot's initial room.

After recognizing NIL the system executes the retrieved skills sequentially. During the execution perceived aspects of the surroundings are recorded in the robot's memory, and the robot can later recall those percepts. Notice that in the example, recalling of the memory is necessary in order that the robot can return into the initial room. In some other cases recalling may be optional. For example, as a result of the example task the robot stores in the memory the fact that a chair is now located in the robot's initial room. A later request to find chair could trigger a recollection of that fact, and the robot could go to check that room first before a search of a chair.

4. SUPERVISED LEARNING IN A SKILL SYSTEM

For each request the robot attempts to retrieve associatively the corresponding skill; the failure in a fetch initiates the teaching cycle. A request is first parsed into its deep case representation of the general form (⟨verb⟩⟨case1⟩⟨case2⟩ ···). For example, the deep structure of (! GO INTO R204) is

((VERB GO) (DESTINATION (PREP INTO) (NOUN R204)))

This request has a single case, DESTINATION; other possible cases are AGENT, OBJECT, TRAJECTORY, INSTRUMENT, and LOCATION. To facilitate associative retrieval, skills are stored as ordered pairs (⟨request pattern⟩, ⟨skill definition⟩). Both request patterns and skill definitions are subject to supervised generalization.

The request pattern of a skill is initially identical to the deep structure of a specific request, namely that whose failed retrieval was resolved by acquiring that new skill. Thus if (! GO INTO R204) was the first instance of requesting the robot to go into *any* named room, a new skill was promted from the master and stored as (⟨go-into-pattern [R204]⟩, ⟨go-into-skill [R204]⟩), where the pattern is the one shown above. Brackets indicate names of applicable objects.

The robot's competence increases through skill generalization and skill acquisition, both under the supervision of the master. Skill generalization focuses on generalizing both request patterns and applicable object names. The former uses a few heuristic rules in rating "closeness" of deep structures and request patterns. A "close" instance calls for the master's opinion, and if he approves the semantic equivalence, the stored request pattern is generalized accordingly. Two requests are considered "semantically equivalent" when a same skill definition (structurally) applies to both.

In the current implementation, requests have been strongly restricted permitting only use of verbs, nouns, and prepositions and similar qualifiers. Request pattern generalization is confined into finding synonyms among verbs and prepositions. For example, if the robot is taught to go into R204 as explained above and it is later requested (! WALK INTO R204), the deep structure of the request and go-into-pattern qualify as being "close" because their single cases, DESTINATION, match. If the master confirms the semantic equivalence, the stored request pattern is modified into:

((VERB (GO WALK)) (DESTINATION (PREP INTO) (NOUN R204)))

The more interesting domain of supervised skill generalization focuses on the ways the robot gets gradually acquainted with objects in its surroundings. In contrast to conventional programming, typology of objects is *not* initially fixed in this case, but rather it develops with the acquisition and generalization of skills. Thus when the robot was taught to go into R204, it learned a skill and a piece of information that R204 is a "go-into-able" object. When it is later taught to apply the same skill to other room names as well, the robot learns the equivalence relation "go-into-able" of room names.

A developing hierarchy called a *logical taxonomy of concepts* (LTOC) represents such an evolving typology of objects. An instance of an LTOC reflects the applicability ranges of so far acquired and generalized skills. The growth of an LTOC is controlled by the attachment of new object names—provided by the master, or new generic names greated by the generalization process—in an inclusion hierarchy.

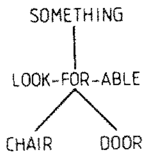

Fig. 1. LTOC after learning (! LOOK FOR DOOR), (! LOOK FOR CHAIR), and (! PICK UP CHAIR).

To give a flavor of the generalization of the applicability ranges of skills, assume a robot possesses only the two skills shown schematically below, and its LTOC is the one in Fig. 1.

```
(((⟨look-for-pattern [LOOK-FOR-ABLE]⟩,
  ⟨look-for-skill-definition [LOOK-FOR-ABLE]⟩)
 (⟨pick-up-pattern [CHAIR]⟩,
  ⟨pick-up-skill-definition [CHAIR]⟩))
```

This state of the robot's competence indicates that the robot has been requested (! LOOK FOR CHAIR), (! LOOK FOR DOOR), and (! PICK UP CHAIR). It is not known in what order and how often the requests have been expressed. Furthermore, the master has, after a promt, confirmed the semantic equivalence of (! LOOK FOR CHAIR) and (! LOOK FOR DOOR), indication that the same skill applies to both, and the skill generalization process has generated a generic name LOOK-FOR-ABLE. At this point the robot "knows" that chairs are objects that are both "look-for-able" and "pick-up-able," but doors are only "look-for-able."

Assuming the robot is then requested (! PICK UP BASKET), the skill retrieval would fail since BASKET is a so far unknown name. The skill generalization phase finds, however, the skill for picking up a chair to be a promising candidate for generalization, since the stored pick-up-pattern matches with the deep structure of the request, save the names CHAIR and BASKET. The master's affirmative answer results in the formation of a new generic name PICK-UP-ABLE. The remaining problem is the reciprocal relationship between LOOK-FOR-ABLE and PICK-UP-ABLE. This is resolved by the master's answer to the question whether a door is a PICK-UP-ABLE. A denial results in the state of skills shown below and LTOC in Fig. 2.

```
(((⟨look-for pattern [LOOK-FOR-ABLE]⟩,
  ⟨look-for-skill-definition [LOOK-FOR-ABLE]⟩)
 (⟨pick-up-pattern [PICK-UP-ABLE]⟩,
  ⟨pick-up-skill-definition [PICK-UP-ABLE]⟩))
```

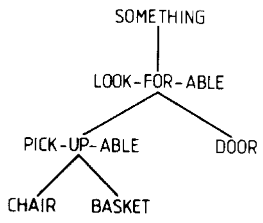

Fig. 2. LTOC after learning (! PICK UP BASKET).

Initially the robot's environment has no structure. Gradually objects begin to "depart" from this chaos when the robot learns to perform actions with them. Generic names engendered in the generalization process come close to what Gibson[4] calls "affordances"; their positions in LTOC indicate what actions their member names "afford."

5. A SKILL SYSTEM

The proposed *skill system* is a quadratuple $SS = (\Phi_S, LTOC, M, I)$, where Φ_S is a evolving collection of skills, LTOC is an evolving logical taxonomy of concepts, M is memory, and I is an interpreter.

As already mentioned, the development of Φ_S and LTOC takes place when the robot does not find a skill for a request. The system then transfers into the supervised learning mode. In that mode the system first attempts to generalize the existing skills with the assistance of the master, and then, if the attempt fails, it asks the master to define a new skill. The skill definition rules are explained in the next section.

The interpreter is composed of several procedures whose task is to receive and parse requests, retrieve the corresponding skills and control their executions. The interpreter also includes procedures for supervised learning, which are implicitly described before.

In this chapter we outline memory and retrieval of its contents. Since the contents of memory is engendered by satisfied "perceptions," a brief discussion on the robot's perceptual faculty shall take place. Perceptions have the dual role in registering perceivable aspects of the surroundings and providing the control function for skills.

The robot's perceptual faculty is composed of a set of distinct perceptual expert procedures, each proficient over a specific sensory invariance. At a call an expert responds TRUE or FALSE depending on whether or not the invariance holds. (SEE TABLE) and (BODY-TOUCH FRONT) exemplify calls of two possible perceptual experts. The first expert returns TRUE if at the moment of a call an image of a table occurs in the visual field; it otherwise returns FALSE. The second expert returns TRUE if the front of the robot's body is touching something.

The robot's attention, when active, is always focused on only those aspects in the surroundings which serve as milestones in the execution of skills. That is, the robot, while executing skills, anticipates certain perceptions and perceives only anticipated facts. Anticipations are indicated by Boolean expressions of calls of perceptual experts and recallings of memory attached as preconditions and goal conditions of skills. When active, the system is thus continuously checking either a goal condition to discontinue a movement or a precondition to initiate one.

The robot's memory records experiences of perceptions, and LTOC records experiences of acts. Perceptual experience is subdivided into the following three functionally distinct parts of memory.

5.1. Perceptual Memory

The robot's perceptual memory faithfully records, as a list, traces of all occurred percepts. A trace of a percept simple represents the call of a perceptual expert which returns TRUE. Traces are recorded in the order of their occurance. For example, the perceptual memory may contain traces:

((BE-IN R204) (SEE TABLE) ···)

5.2. Propositional Memory

The robot's propositional memory contains various propositions abstracted from the contents of the perceptual memory. Abstractions are performed by expert procedures which independently watch conditions for abstraction in the perceptual memory. For example, the procedure for abstracting the property "location" would abstract and record in the propositional memory from the previous example the following proposition:

(TABLE LOCATION R204)

5.3. Orienting Map

Spatial organization of the robot's environment is recorded in an orienting map. Assuming a house equals the robot's world, the adjacency relation of the rooms and an indication of their floor locations constitute a plausible orienting map; incidentally, such a map was used in the simulation.

For example, such an orienting map may then contain entries of the following kind:

(R204 ADJACENT-TO (R205 R201))
(R204 FLOOR 1)

5.4. Logical Taxonomy of Concepts

LTOC reflects the robot's action-based understanding of objects in the surroundings as already explained. The robot only "knows" to the extent it can do, and the more it is able to act on an object the more it "knows" about it. Initially, the robot is familiar with a single generic concept name SOMETHING. From that seed LTOC emerges under the influence of two complementary forces: one for breaking existing generic concept names under new created ones and one for gathering names under higher level ones. Figure 3 shows an advanced LTOC. Generic names (the ones between the "root" and the "leaves" of LTOC) indicate how acquired skills apply to known objects. $ROOM is the name of a reference variable used in indirectly referring to objects.

Contents of memory and LTOC are retrieved in a uniform manner by calling function RECALL with a proper set of arguments. The role of recalling is similar to that of perceiving, viz. setting preconditions in skills. The basic form of recalling is:

(RECALL ?⟨reference variable⟩ [⟨intension1⟩
[EXCEPT ⟨intension2⟩]])

Its interpretation is: return those names of objects in LTOC which are siblings or offsprings of siblings of the reference variable and (optional) which satisfy intension1, but (optional) reject the ones that satisfy

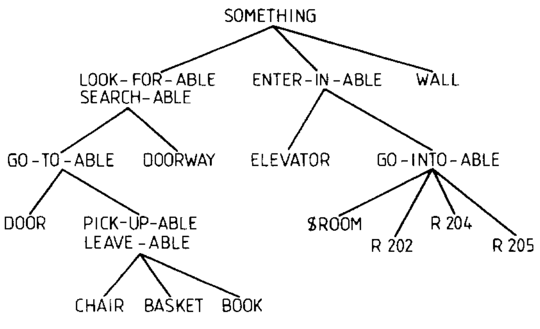

Fig. 3. An advanced LTOC.

intension2. Recalled names will be bound to the reference variable as a list. If the recalled list is empty, the call evaluates FALSE, otherwise it evaluates TRUE. Below appear a few example recalls. Notice how indirect recalling is established by referring to an already "bound" reference variable:

(RECALL ?ROOM BE-IN FIRST)
Recall the room in which you were first

(RECALL ?ROOM ADJACENT-TO R202)
Recall the rooms adjacent to R202 (as indicated in the Orienting Map)

(RECALL ?ROOM ADJACENT-TO $ROOM)
Recall the rooms adjacent to the rooms reference variable $ROOM refers to.

6. THE STRUCTURE OF SKILLS

Three types of complex skills, each hierarchically defined using lower-level skills or primitive skills are admissible. A primitive skill represents a movement coupled with a goal state recognizer—a Boolean expression of calls of perceptual experts—to discontinue the movement. In defining a skill, types can be freely mixed. The syntactic rule, its semantic interpretation, and an example are shown below for each skill type. The examples are drawn from the simulation. Notice that if a constituent lower-level skill does not obey the syntactic form of primitive skills, it refers to complex constituent skill available through a parse-fetch cycle. A precondition represents a boolean experession of calls of perceptual experts and/or recalls. AND-operation is defaulted on the top level of a precondition.

[PS] (⟨movement⟩ UNTIL ⟨goal state⟩)
"Discontinue the (activated) movement when the goal state evaluates TRUE."

Ex4: (MOVE-FORWARD UNTIL (BODY-TOUCH FRONT))

[S1] (⟨precondition⟩ → ⟨defining skill⟩⟨defining skill⟩ ···)
"If the precondition evaluates TRUE, perform the defining skills sequentially."

```
Ex1: ; skill for going into an adjacent room to a named room
     ; 'recall the adjacent rooms and go into a closest one'
      (((RECALL ?ROOM ADJACENT-TO GO-INTO-ABLE))
          → (GO INTO $ROOM))
```

[S2] (EXCLUSIVE
⟨precondition⟩ → ⟨defining skill⟩⟨defining skill⟩ ···

```
        ⟨precondition⟩ → ⟨defining skill⟩⟨defining skill⟩ ···
         ···)
```

"Choose that one of exclusive skills whose precondition evaluates TRUE. Perform it as if it were a S1-skill."

```
Ex2: ; skill for going into a named room,
     ; triggered by a specific request of the general form:
     ; (! GO INTO GO-INTO-ABLE)
     ; 'if you already are in the room then o.k.,
     ; otherwise if you are on the same floor
     ; and in an adjacent room, go to the
     ; separating door, open the door, and enter'
       (EXCLUSIVE (((BE-IN GO-INTO-ABLE)) → (DONE))
                  (((NOT (BE-IN GO-INTO-ABLE))
                    (BE-ON-FLOOR-OF GO-INTO-ABLE)
                    (BE-IN-ADJ-TO GO-INTO-ABLE))
                    → (GO TO GO-INTO-ABLE-DOOR)
                      (OPEN GO-INTO-ABLE-DOOR)
                      (ENTER GO-INTO-ABLE))
```

[S3] (REPEAT-UNTIL ⟨goal state⟩⟨defining skill⟩⟨defining skill⟩ ···)
"Until the goal state evaluates TRUE, repeatedly perform the defining skills."

```
Ex3: ; skill for searching a named object,
     ; triggered by a specific request of the general form:
     ; (! SEARCH SEARCH-ABLE)
     ; 'until you either see the object or
     ; no rooms are left, remember other rooms,
     ; go into a closest one, and look for the object there'
       (REPEAT-UNTIL (OR (SEE SEARCH-ABLE)
                         (RECALL $ROOM))
                 (REMEMBER OTHER ROOMS)
                 (GO INTO $ROOM)
                 (LOOK FOR SEARCH-ABLE))
```

In [PS] or [S3] a goal condition, by definition, is attached to a movement or lower-level skills which drive the robot to a full evaluation of the condition. Since preconditions are not connected to such built-in behavioral solutions, general skills, called *perceptual expectations*, are reserved to dynamically solve the subproblems of unfulfilled preconditions. They perform such bodily movements that anticipated but unsatisfied preconditions would evaluate TRUE. The system automatically checks the preconditions when a new skill is defined and asks for definitions for all unknown perceptual expectations.

For example, Ex1 may represent the perceptual expectation of (BE-IN-ADJ-TO GO-INTO-ABLE) perception used in Ex2. If the robot is requested to go into a named room, Ex2 is activated and the general concept name GO-INTO-ABLE is replaced with the requested room name. Should the robot be located on the same floor but not in an adjacent room to the target room, Ex1 will be called for execution causing Ex2, in this case, to recursively call itself.

The use of perceptual expectations represents an implementation of a *backward chaining* execution of ⟨situation⟩ → ⟨action⟩ rules. A chaining can automatically expand itself because an unsatisfied precondition of a triggered perceptual expectation in turn activates other expectations and so forth. In contrast to customary backward chaining of production rules, there is *no* search involved in this backward chaining execution of skills, because a retrieval mechanism directly finds means (perceptual expectations) for unsatisfied situations (preconditions).

7. ON IMPLEMENTATION AND SIMULATION

A robot system as envisioned above has been implemented and its performance simulated in a fictional environment. The system itself was programmed in MTSLIPS language on an IBM S/370 M168 computer; the simulated world was programmed in FORTRAN language on a Tektronix 4010 graphics terminal. The overall program, for the most part comprising a LISP interpreter, FORTRAN graphics routines, and skills, runs in about 190 k bytes.

The system has two distinct modes of performance: (i) in the interactive task definition and teaching mode a master requests tasks and advises the system, and (ii) in the execution mode the robot performs tasks. Figure 4 depicts these two modes of performance.

The master expresses a task as a list of requests, imperative natural language sentences. A parser first maps a request into its deep case structure, and then the retriever attempts to fetch a matching skill definition. The retriever accepts the first skill definition whose request pattern part matches the request, and then replaces all generic names in the skill with their instances in the request. If the retriever does not catch any skill, control is passed to the skill generalization module.

The generalization module walks through the skills one by one in a search for a skill whose pattern would be "close" enough to the request to qualify as a candidate for generalization. Several heuristic rules are used for determining closeness. A necessary condition is a match between the deep

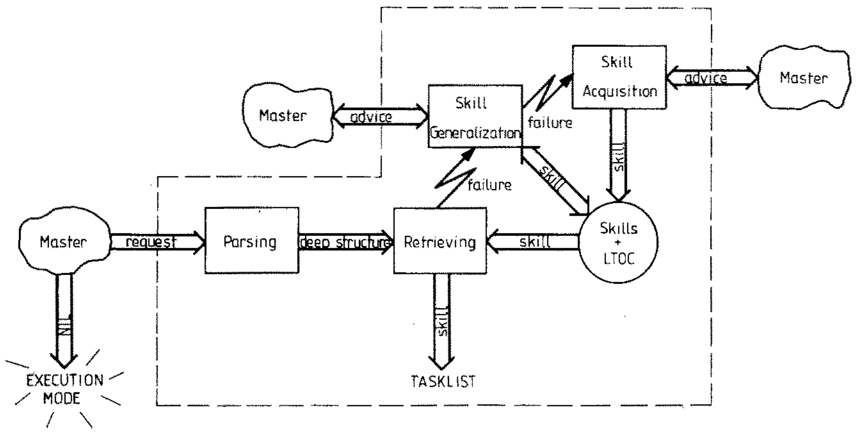

Fig. 4. Schematic description of the skill system.

cases of the request and the pattern. Any promising candidate is displayed for confirmation. The master's acceptance results in the modification of the skill (and possibly LTOC). A failure passes control to the skill acquisition module. The acquisition module asks the master to define a new skill and displays the existing skills and the rules for well-defined skills. A new skill is appended to the tasklist and stored for future use.

The master's NIL terminates a task definition and initiates the execution mode. In this mode the retrieved skills, organized as a list, are executed serially, and perceived facts of the surroundings are recorded in the robot's memory. Skills are hierarchically composed of lower-level skills; basic building blocks are primitive skills. Primitive skills are directly indicated in a skill definition and can thus readily executed. Higher-level component skills are indicated by macro names (generalized surface requests). The execution of a retrieved skill forms thus a sequence of parse-fetch cycles which is controlled by perceptions and recalling of memory and occasionally interrupted by the executions of primitive skills.

The performance of a skilled robot in a house was simulated on a Tektronix 4010 graphics terminal. The simulated house comprises three floors with six rooms on each and an elevator connecting the floors. The rooms contain randomly distribured tables, chairs, and baskets. The robot manipulates those objects as requested.

Any implementation of a skilled robot system would require a careful design and implementation of task-dependent motor movements and perceptual expert procedures. They form a base for subsequent skill structures. Table I exhibits the motor movements of the simulated robot. Since the simulation displays the robot's movements on the graphics

Table I. Motor Movements of the Simulated Robot

MOVE-FORWARD
MOVE-BACKWARD
MOVE-RIGHT
MOVE-LEFT
TURN-RIGHT
TURN-LEFT
WAIT

Table II. Some Perceptual Experts of the Simulated Robot

(SEE ⟨object⟩)
(SEE ⟨room⟩-DOOR)
(SEE ⟨object⟩ IN-FRONT-OF ⟨object2⟩)
(BE-IN ⟨room⟩)
(BE-IN-ADJ-TO ⟨room⟩)
(BE-ON-FLOOR-OF ⟨room⟩)

Table III. Elementary Skills of the Simulated Robot

(GRASP-BACK ⟨pick-up-able⟩)
(GRASP-FRONT ⟨pick-up-able⟩)
(UNGRASP-BACK ⟨pick-up-able⟩)
(UNGRASP-FRONT ⟨pick-up-able⟩)
(OPEN ⟨door⟩)
(CALL-ELEVATOR)
(PUSH-BUTTON ⟨floor-no⟩)

```
>>> GIVE YOUR FIRST SENTENCE NOW.
    (! GO INTO R202)
   *OK. AND THEN?
    (! TAKE TABLE)
   *OK. AND THEN?
    (! GO INTO R204)
   *OK. AND THEN?
    (! LEAVE TABLE BY WALL)
   *OK. AND THEN?
    NIL
```

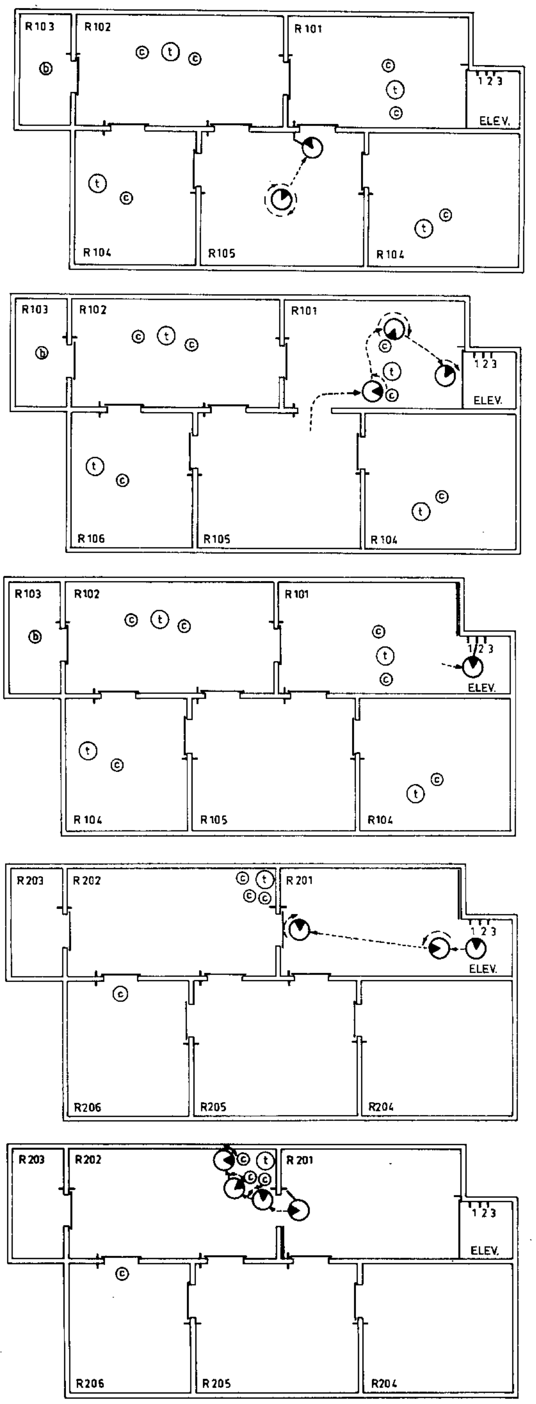

Fig. 5. An example task performance.

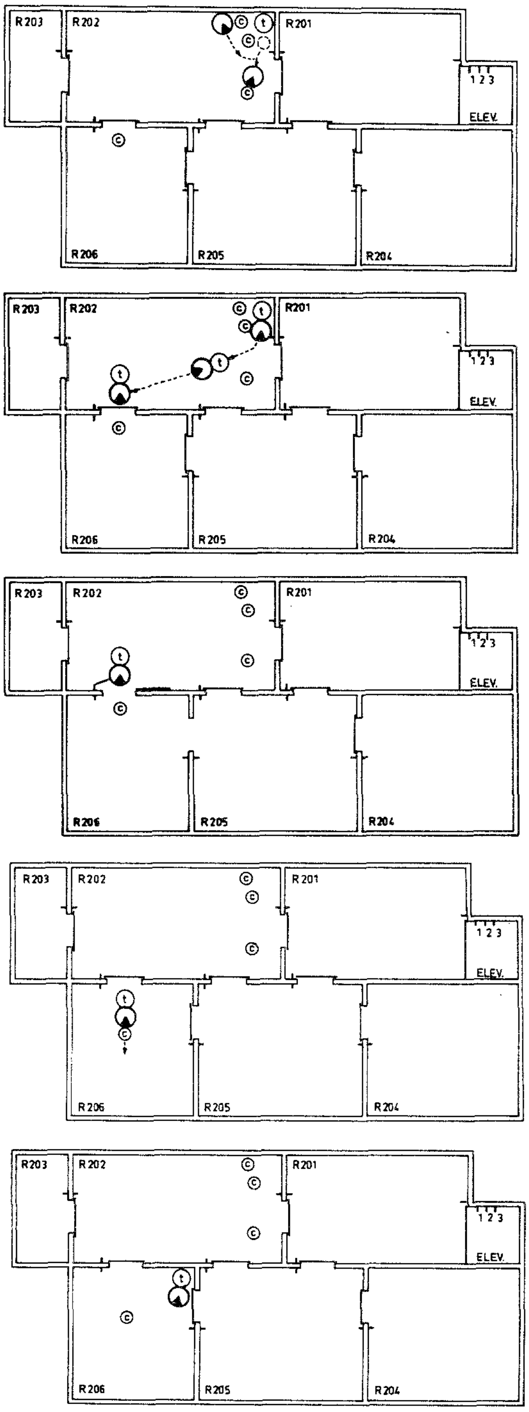

Fig. 5. Continued.

terminal, the simulated implementations of the movements correspond to respective displacements on the screen.

Table II shows some of the implemented perceptual expert procedures. The simulated perceptual strategies necessarily differ radically from and have been often a great deal easier to implement than most real implementations. Particularly, the simulated vision is trivial in comparison with the corresponding real scene analysis procedures. (BE-IN ⟨room⟩) calls a perceptual expert for recognizing whether or not the robot is located in the room indicated in a call. This expert could in a real implementation be transformed into a use of scene analysis by, say, marking the rooms with discriminative colors. Similarly, visually recognizing individual doors could be greatly alleviated by marking the doors with colors of the rooms they are leading to.

In the simulation hypothetical detailed hand manipulations were evaded by implementing a set of elementary skills directly on the level of graphics routines. The robot was conceived as possessing two pairs of hands: one pair on the back for carrying objects, the other pair on the front for clearing the way, pushing buttons, and so forth. Table III exhibits the implemented elementary skills.

The simulation demonstrated that complex flexible behavior over a nontrivial set of tasks can be generated by a small collection of general skills. In the simulation the robot in its most developed stage possessed 28 general skills, the average complexity being five or so defining skills. The example skills of Sec. 6 represent typical simulated skills. The skills endowed the robot with the general competency to perform, among others, the task of Fig. 5 regardless how the robot was initially located and what obstacle existed on its way. The robot dynamically decided when to use the elevator.

In another simulation instance the robot was requested to find a basket:

```
(! FIND BASKET)
(! PICK UP BASKET)
(! COME BACK)
(! LEAVE BASKET BY WALL)
NIL
```

Such a task fits well in the perception-controlled skill formalism. After checking that a basket was not located in the initial room, the robot recalled all rooms in the house and visited them one by one until it found a basket. When the robot was later requested to find a basket, it recalled the location of the previous basket and went straight to pick it up.

In still another instance the robot was taught that "bringing" an object to a target room means finding such an object in a room other than the target room and then carrying the object into the target room. The robot was then able to perform the following task:

```
(! BRING CHAIR INTO R106 BY TABLE)
NIL
```

Jäppinen(7) contains more pictorial simulation results.

8. SUPERVISED LEARNING REVISITED

Together with the use of "perceptions" for run-time control of behavior, the other thrust of a skill system (SS) is to utilize supervised learning for increasing incrementally a robot's knowledge of the world in action form. Since the teaching method of SS resembles supervised learning device in Tangwongsan and Fu(13) (T&F), it is of interest to compare the two strategies.

Both systems employ a teaching strategy which might be called "learning by imitation" (through verbal means, to distinguish it from more common learning by imitation where a trainee perceives actions of a competent trainer). Learning by imitation constitutes a large portion of the acquisition of knowledge in action form particularly for the child but also for the adult. An apprenticeship represents a contractual assignment to utilize learning by imitation. The strategy is valuable vehicle for transferring advantageous behavioral patterns through generations, since a trainee does not have to perform inferrence to learn those patterns—it is sufficient to be receptive. A great many of human skills, such as playing a musical instrument, tying shoelaces, driving a car, etc., have been acquired in this manner.

Both T&F and SS choose imperative natural language sentences for expressing tasks for a robot. This is a deviant notation from traditional AI robotics where tasks are expressed as goal states of a task world. Imperative sentences are convenient and fit well in describing processes, while goal states express well end results of processes but fit poorly in describing processes themselves. For example, it is hard to express a juggling task as goal states, while imperative sentences signal the task unambiguously. T&F is more ambitious in task expressions than SS by allowing unrestricted sentences whereas SS requires task definitions be broken into lists of single clause sentences. Both systems choose the case grammar for expressing semantic deep representations of sentences. Deep case structures of imperative sentences are used in both systems for associating task expressions with learned action propensities.

The main difference between the two systems deals with the nature of acquired (taught) action propensities. While T&F adheres to the working-horse of AI robotics that a robot executes *plans*—operator sequences which refer to coordinate points in a world model—SS uses *skills* which refer to actual states of the environment through the act of "perceiving." Consequently, acquired action propensities of T&F are more specific and call for greater degree of abstraction to be usable in other tasks. Skills are more general pruning the need for abstraction. For example, a plan for the task "Go to a big box in Room6" in T&F defines operators for going from the initial coordinate point of the robot to a door, opening the door, going through it, closing it, and going to the coordinate point of a big box. (When the box was located in an adjacent room.) To apply the plan for analogous tasks T&F has to abstract away positions of the robot and a box. In SS the identical task is expressed as ((! GO INTO ROOM6) (! GO TO A BIG BOX) NIL). The two skills they trigger carry no information about positions. The first one will ask to look for a door, to go to it, to open it, and to go through it. The other skill employs similar strategy.

SS performs an important form of abstraction in gradually building a taxonomy of manipulable objects while a robot learns new skills and is introduced new objects. A given stage of the taxonomy, then, explicitly defines semantics of objects operationally by indicating what actions they "afford." The robot learns hand-in-hand both skills and semantics of objects in supervised learning. It seems that T&F does not consider this learning facet, and semantics of objects is initially given and remain fixed.

Finally, plans of T&F are clearly more powerful (or less confined) than skills of SS. Skills receive control information through senses, and the "reach" of any sense organ is necessarily geographically limited. Plans of T&F do not suffer such limitations since their control rely on a world model. For the task "Bring a chair into the current room," the robot of T&F would find from its world model the location of the closest chair, but the robot of SS has to trigger a search skill to guide the robot into rooms one after another in search of a chair. Only by luck a found chair is the closest one.

9. CONCLUSION

To obtain a robot system whose response would be fast *and* flexible we have designed, implemented, and simulated a skilled robot. Since the robot performs almost no inference after receiving a task definition,its response is fast, in fact almost immediate. The robot relies minimally on a world model (only in orienting itself), and, accordingly, the robot's behavior is flexible even in a changing environment.

Using perceptions for control and imposing hierarchy, skills can be defined to strongly capture regularities in behavior and, consequently, only a small number of skills is sufficient in nontrivial task domains, as the simulation demonstrated.

Undoubtedly, the greatest drawback of a skilled robot is its lack of inference. Albeit skills do generate correct (but not always optimal) behavior for tasks even in complex worlds, they do so in manner imposed on their general structures. At any moment in an ongoing behavior, a skill controls the behavior based on only the temporally *present* (or *past*) and geographically *near* states of the world. But in no way can *future* or geographically *distant* states influence the behavior. A skilled robot would never insert a coin in a piggy bank for a future use. Compared with inferred behavior, skilled behavior gains in efficiency and accuracy of response but loses in the predictive power. How severe a confinement this disadvantage represents does not depend on the complexity but rather on "regularity" of a task domain.

We are currently looking for ways of implementing a planning mechanism together with a world model *on* a skill base. Such a system would combine the best of skilled and inferred behavior.

REFERENCES

1. S. E. Fahlman, "A planning system for robot construction tasks," *Artif. Intell.* 5:1–49 (1974).
2. R. E. Fikes, P. E. Hart, and N. I. Nilsson, "Learning and executing generalized robot plans," *Artif. Intell.* **3**:251–288 (1972).
3. C. J. Fillmore, "The case for case," in *Universals in Linquistic Theory*, E. Bach, and R. Harms, eds. (Holt, Rinehart, and Winston, 1968), pp. 1–88.
4. J. J. Gibson, "The theory of affordance," in *Perceiving, Acting, and Knowing*, R. Shaw, and I. Bransford, eds (Lawrence Erlbaum, 1977), pp. 67–82.
5. P. I. Hayes, "A representation for robot plans," *4th International Joint Conference on Artificial Intelligence* (1975), pp. 181–188.
6. H. Jäppinen, "Heuristic programming—an emerging trend in the 80's," *Proceedings of NordDATA-80 Conference* (1980), pp. 551–556.
7. H. Jäppinen, "An application of teaching to intelligent robot behavior," Helsinki University of Technology, Digital Systems Laboratory, Report B9, (1980).
8. H. Jäppinen, "Intelligent behavior without planning," *The Fifth European Meeting on Cybernetics and Systems Research* (1981).
9. J. Piaget and B. Inhelder, *The Psychology of the Child* (Basic Books, New York, 1969).
10. E. D. Sacerdoti, *A Structure for Plans and Behavior* (American Elsevier, New York, 1977).
11. E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artif. Intell.* 5:115–135 (1974).
12. G. Sussman, *A Computer Model of Skill Acquisition* (American Elsevier, New York, 1975).
13. S. Tangwongsan and K. S. Fu, "An application of learning to robotic planning," *Int. J. Comput. Inform. Sci.* **8**(4):303–333 (1979).