SRI International

A FUZZY CONTROLLER FOR FLAKEY, AN AUTONOMOUS MOBILE ROBOT

Technical Note No. 529

March 1993

By: Alessandro Saffiotti†, International Fellow
Enrique H. Ruspini, Sr. Computer Scientist
Kurt G. Konolige, Sr. Computer Scientist
Artificial Intelligence Center
Computing and Engineering Sciences Division

† Currently at: IRIDIA, Universite Libre de Bruxelles, Brussels, Belgium

The first author was supported by a grant from the National Council of Research of Itally. Research performed by the second author, leading to the conceptual structures used in the autonomous mobile vehicle controller, was supported by the U.S. Air Force Office of Scientific Research under Contract No. F49620-91-C-0060. Support for Kurt Konolige was partially provided for by the Office of Naval Research under Contract No. N00014-89-C-0095., and by SRI International.

Abstract [1]

Controlling the movement of an autonomous mobile robot in real-world unstructured environments requires the ability to pursue strategic goals under conditions of uncertainty, incompleteness, and imprecision. We describe a fuzzy controller for a mobile robot that can take multiple strategic goals into consideration. Through the use of fuzzy logic, goal-oriented behavior (e.g., trying to reach a given location) and reactive behavior (e.g., avoiding previously unknown obstacles on the way) are smoothly blended into one sequence of control actions. The fuzzy controller has been implemented on the SRI robot Flakey, and its performance demonstrated in several different environments, including the first AAAI robotic competition, where Flakey placed second.

[1]Excerpts from this report appeared in the Second IEEE Conference on Fuzzy Systems (San Francisco, CA, March 1993), and in the Fuzzy Logic '93 Conference (Burlingame, CA, July 1993).

Contents

1 Introduction

Autonomous operation of a mobile robot in a real environment poses a series of problems. First, knowledge about the environment is, in general, incomplete, uncertain, and approximate. For example, maps typically omit some details and temporary features, spatial relations between objects may have changed since the map was built, and the metric information may be imprecise and inaccurate. An autonomous robot must be able to make effective use of available information regardless of its inconvenient characteristics. Second, perceptually acquired information is usually unreliable. Noise in sensor measurements introduces uncertainty; their limited range, combined with the effect of environmental features (e.g., occlusion), leads to imprecise and incomplete data; and errors in the measurement interpretation process may lead to incorrect beliefs. Robot control, based on an intelligent combination of prior information and current observations, should be robust and tolerate errors associated with observations and their interpretation. Real-world environments also have complex and unpredictable dynamics: objects can move, other agents can modify the environment, and relatively stable features may change slowly with time (e.g., seasonal variations). An autonomous robot must be able to detect unexpected events and promptly react to them (e.g., a person walking in front of the robot). Finally, the effect of control actions is not completely reliable. In general, the results produced by a control command can be only approximately estimated and, sometimes, the command may just fail. The behavior of the robot should not critically depend on the precision of the effectors, and failures should be readily recognized and accounted for. In addition, decisions and actions must take place in real time. New goals, or unexpected events, may require immediate attention—proving theorems about what to do may not be wise when a truck is approaching.

Classical planning approaches to the control of mobile robots have been criticized for not providing the real-time responsiveness, or *reactivity*, that is needed in real-world environments (unexpected contingencies must be dealt with by reinvoking the planner, usually an extremely costly operation), and for being too inflexible in the face of uncertainty and imprecision in the information used, and errors in the execution [Firby, 1987; Kaelbling, 1987; Gat, 1991; Saffiotti, 1993]. To overcome these limitations, some authors have proposed architectures centered around a complex controller, or *reactive planner* (e.g., [Firby, 1987; Kaelbling, 1987]); some of these proposals have been based on the use of fuzzy control techniques (e.g., [Sugeno and Nishida, 1985; Yen and Pfluger, 1992]). These reactive architectures provide immediate response to unpredicted environmental situations by giving up the idea of reasoning about future consequences of actions. Reasoning about future consequences (sometimes called "strategic planning"), however, is still needed in order to intelligently solve complex tasks (e.g., by deciding not to carry an oil lantern downstairs to look for a gas leak [Firby, 1987].)

One solution to the dual need for strategic planning and reactivity is to adopt a two-level model: at the upper level, a planner generates a sequences of operations whose performance is expected (in an ideal world!) to satisfy the robot's goals; at the lower level, a controller tries to achieve these goals while dealing with the environmental contingencies. This solution requires that the controller be able to accept a specific *goal* as a parameter — this contrasts with typical controllers, including fuzzy controllers, which are normally defined to achieve one specific, predetermined functionality of the system. Moreover, the controller must be able to consider several such goals simultaneously — for example, to (1) traverse a hallway while (2) avoiding the obstacles and people on the way and (3) tracking a specific landmark with its camera. A major problem in the design of such a controller is how to resolve conflicts between simultaneous goals.

In this paper, we describe a reactive controller for an autonomous mobile robot that uses fuzzy logic for trading off conflicting goals. Goals are either "innate", like avoiding collisions, or "strategic", communicated by a planner or by a human programmer. This controller has been implemented on Flakey, the mobile robot platform of the Artificial Intelligence Center of SRI International; it implements robust high-level robot actions (like traversing a hallway, or crossing a door), and provides capabilities for

- Robust, uncertainty-tolerating goal-directed activity
- Real-time reactivity to unexpected contingencies
- Blending of multiple goals

The scope of the techniques we describe extends, however, beyond this particular test-bed, as they effectively deal with basic problems in the development of intelligent autonomous systems such as the attainment of multiple, possibly conflicting, objectives; the integration of numerical control and planning techniques based on symbolic reasoning; and the construction and utilization of approximate and incomplete models of the world.

The formal bases for the proposed controller have been set forth by Ruspini [Ruspini, 1990; Ruspini, 1991a] after the seminal works by Zadeh (e.g., [Zadeh, 1978]). In a nutshell, each goal is associated with a function that maps each perceived situation to a measure of desirability of possible actions from the point of view of that goal. The notion of a *control structure* is used for introducing high-level goals into the fuzzy controller. Intuitively, a control structure is an object in the robot's workspace, together with a desirability relation and a context of applicability: typical control structures are locations to reach, walls to follow, doors to enter, and so on. Each desirability function induces a particular behavior — one obtained by executing the actions with higher desirability. Behaviors induced by many simultaneous goals can be smoothly blended by using the mechanisms of fuzzy logic. In particular, reactive and goal-oriented behaviors are blended in this way into one sequence of control actions. This compositional approach to complex behavior is formally explored in deeper detail in a companion technical report [Saffiotti *et al.*, 1993a].

The next section gives a brief overview of Flakey and outlines its present architecture. Section 3 sketches the architecture of the controller, and describes the way reactive behaviors are implemented. Section 4 deals with the introduction of high-level goals into the reactive controller. Section 5 attacks the problems of blending different behaviors, and presents several illustrative experiments run on Flakey. Finally, Section 6 discusses the results and conclusions.

2 The mobile robot test-bed

Flakey is a custom-built mobile robot platform approximately 1 m high and 0.6 m in diameter for use in an indoor environment. There are two independently driven wheels, one on each side, giving a maximum linear velocity of about 0.5 m/s. Flakey's sensors include a ring of 12 sonars, giving information about distances of objects up to about 2 m; wheel encoders, providing information about current linear and rotational velocity; and a video camera, currently used in combination with a laser to provide dense depth information over a small area in front of Flakey. A passive-vision system is currently being added. In addition, Flakey has enough on-board computational power to run all the low-level and high-level interpretation and control processes; high-level processes can also be run remotely through a radio link for better programming and debugging convenience.

Figure 1 illustrates the part of Flakey's architecture that is relevant to the controller. The sensorial input is processed by a number of interpretation processes at different levels of abstraction and complexity, and the results of interpretation are stored in the *local perceptual space* (LPS). The LPS represents a portion of a Cartesian plane, centered on the robot, containing information about its vicinity. The LPS integrates information coming from the sensors at different levels of abstraction and interpretation (represented by different levels of gray in the figure). In Figure 1, points corresponding to surfaces identified by the sonars and the camera are visible in the LPS — Flakey is the octagon in the middle of the LPS, in top-view. The line on the left is a *wall hypothesis* built by the sensor interpretation processes. The *fuzzy controller* can use any of the objects represented in the LPS at input: this input is checked every 100 ms, and a corresponding control action is generated.

The LPS also integrates information coming from the planner, in the form of *control structures*, abstract representations of local strategic goals to be achieved by the robot. Intuitively, a control structure stands for an object in the robot's workspace together with a fuzzy goal predicate involving this object. For example, the circle near the upper left corner of the LPS in Figure 1 is a control structure: it consists of a *position*, and the fuzzy predicate at measuring how much Flakey is *at* that position.

A set of control structures may be used to express a decomposition of a complex

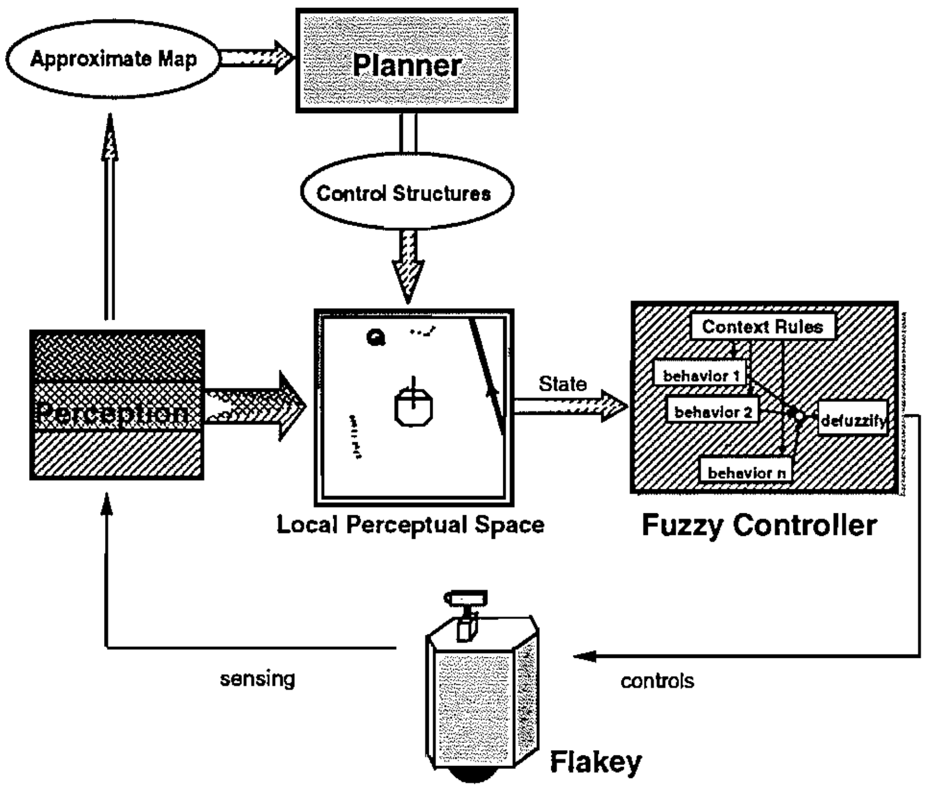


Figure 1: A simplified view of Flakey's architecture.

task into basic subtasks. These structures can be directly specified by humans or generated automatically by standard AI planning or decision-making techniques. In our current experiments, we use a simple *planner* to generate a set of control structures that achieves a prespecified navigational goal. This planner uses a sparse topological *map* of the environment annotated with approximate metric information. When input to the LPS, control structures induce a bias over the reactive execution of the fuzzy controller. More details on our approach to the integration between planning and fuzzy control can be found in a companion technical report [Saffiotti *et al.*, 1993a].

The fuzzy controller regulates the execution of basic control activities called *behaviors*. Intuitively, each behavior is one particular control regime that focuses on achieving one specific, predetermined goal, like avoiding obstacles, or reaching a specific location (given as a parameter). Hence, each behavior can be thought of as being one separate parametric fuzzy controller, implementing a particular skill. Behaviors take their input from the LPS: behaviors intended to promote quick reactivity (e.g., obstacle avoidance) utilize low-level perceptual data, while behaviors intended to promote purposeful actions rely on control structures as their main input. The next two sections are devoted to an analysis of our solution to implement reactive and purposeful behaviors, respectively.

Usually, several behaviors are simultaneously active in the fuzzy controller. For example, at some time, a behavior for traversing a corridor, another to avoid obstacles, and a third one to facilitate door detection, may all be considered by the controller to determine the adequacy of each possible control action. Multiple, possibly conflicting, behaviors are blended into a dynamic sequence of control actions by means of fuzzy logic techniques. These actions are the result of trade-offs between the goals associated with each active behavior. Before the behaviors are combined, the output of each behavior is weighted according to how much that behavior applies to the current contextual situation — for example, a behavior for following a wall is scarcely applicable in the situation where there is an obstacle standing in front of Flakey, while one for avoiding obstacles is more adequate. The *context rules* encode the information about the applicability condition of each behavior. Behavior blending is the subject of Section 5.

3 Reactive behavior

Each *behavior* in the fuzzy controller is responsible for producing a certain type of movement, aimed at achieving or maintaining a particular goal — for example, to stay away from obstacles. Desirable behavioral traits are expressed as quantitative *preferences* over possible control actions from the perspective of the goal associated with that behavior. For example, a behavior for avoiding obstacles could map configurations of sonar readings that correspond to the presence of an obstacle on the left of the robot into a function that prefers actions that steer the robot to the right. Behaviors like this one, directly mapping sensor configurations in the LPS to a preference over possible control actions, are called *reactive behaviors*: these are the most basic form of behavior that Flakey can exhibit.

In more formal terms, and following the semantic characterization of Ruspini [Ruspini, 1991a; Ruspini, 1991b], we describe each behavior B in terms of a desirability function

(1) $$Des_B : \text{State} \times \text{Control} \rightarrow [0,1]\,,$$

that measures, for each state s (i.e., configuration of the LPS) and for each control vector c, the desirability $Des_B(s,c)$ of applying the control c when the state is s *from the point of view of* B. Equivalently, we can say that Des_B associates each situation s with the fuzzy set $\tilde{C}$ of control values characterized by the membership function $\mu_{\tilde{C}}(c) = Des_B(s,c)$. In general, c is an n-dimensional vector of values for all the control variables; in the case of Flakey, c is a pair (α, θ), where α is a linear acceleration and θ is a turning angle.

It is important to notice that desirability functions are less committal than classical control functions — or similar formal objects used in other approaches to robot control, like Khatib's pseudo-potentials [Khatib, 1986] of Arkin's motor schemas [Arkin, 1990]). In fact, desirability functions do not map input states to effector commands (or controls), but to a measure of desirability over the space of these commands. The intuition behind this is that many alternative commands can generate, to a greater or lesser extent, the same *type* of movement, or behavior. This has been made precise by formalizing behaviors in the framework of multivalued logic.

In practice, each behavior is implemented in Flakey by a fuzzy machine struc-

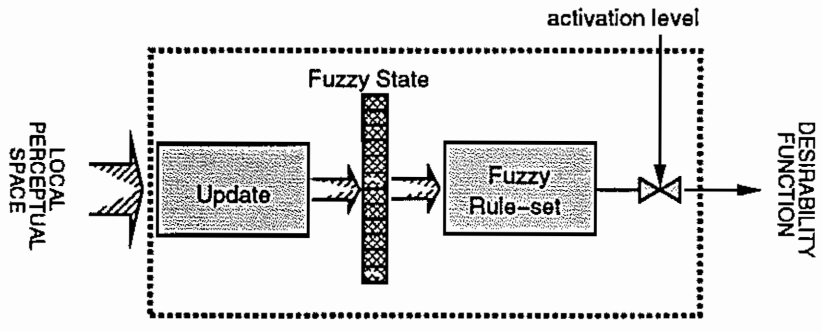


Figure 2: Implementation of a behavior.

tured as shown in Figure 2 (the "activation level" is discussed later). The *fuzzy state* is a vector of fuzzy variables (each having a value in $[0, 1]$) representing the truth values of a set of fuzzy propositions of interest (e.g., "obstacle-close-on-left"). At every cycle, the **Update** module looks at the (partially) interpreted perceptual input stored in LPS, and produces a new fuzzy state.

The **Fuzzy Rule-Set** module contains a set of fuzzy rules of the form

$$\text{IF } A_i \text{ THEN } C_i\,, \tag{2}$$

where A_i is a propositional formula in fuzzy logic incorporating fuzzy predicates over state variables (and possibly the fuzzy connectives AND, OR, and NOT), and C_i is a fuzzy set of control values: for any possible control action c, $C(c)$ tells how much c is a good instance of C. When evaluating the antecedent A_i, we use max, min, and complement to 1 to compute the truth value of disjunction, conjunction, and negation, respectively.[2]

For example, Flakey includes a KEEP-OFF behavior, intended to keep it safely away from occupied areas (obstacles). This behavior includes the following rule R:

[2]Throughout this paper we use the standard min t-norm ($\circledast$), and the correspondent max t-conorm ($\oplus$) [Schweizer and Sklar, 1983]. These are the operations used in the actual implementation.

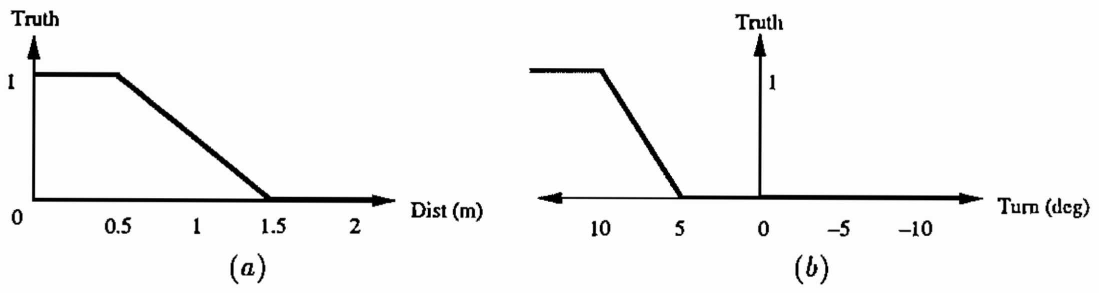


Figure 3: The fuzzy predicates "obstacle-close" (a), and "sharp-left" (b).

```
            IF obstacle-close-in-front AND NOT obstacle-close-on-left
THEN turn sharp-left
```

Figure 3 shows the truth value of the "obstacle-close" predicate as a function of the distance d of the closest object detected by Flakey's sensors, and the "sharp-left" predicate as a function of the turning angle. Consider the obstacle avoidance rule above, and suppose a state s where the sonar sensors detect some object in front of Flakey at 0.7 m. Then, the above rule R would generate the following desirability function D_R:

$$\begin{aligned} D_R(s,5) &= \min(0,0.8) &= 0 \\ D_R(s,7.5) &= \min(0.5,0.8) &= 0.5 \\ D_R(s,10) &= \min(1,0.8) &= 0.8\,. \end{aligned}$$

where we are only considering the value of the θ (turning) control variable.

The rule set in a behavior B normally includes several rules like (2), each expressing a set of desirable controls to use when the state s satisfies the antecedent. Each rule R_i in the set computes a corresponding desirability function D_i as above; these are then unioned using the $\oplus$ (max) operator to obtain an overall desirability function

$$Des_B(s,a) = D_1(s,a) \oplus \ldots \oplus D_n(s,a)$$

It is the responsibility of the design engineer to assure that Des_B is an adequate approximation of the intended desirability function for the specific behavior B, that is, that the rules produce the expected effect. An interesting problem is what guidance, if any, the theory can give us on how to generate sets of rules that produce a given control. This problem is a subject of our current study [Ruspini and Saffiotti, 1993].

Once a desirability function $Des_B(s,c)$ has been computed for the current state s for each control c, the fuzzy controller needs to select one single preferred control action $\hat{c}$ to send to the effectors for execution. This choice is made using centroid defuzzification, that is,

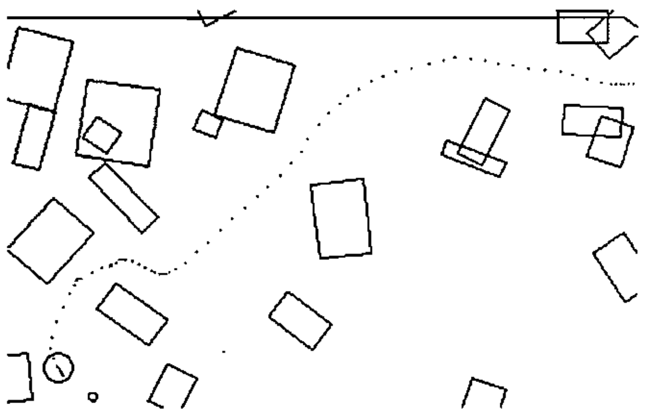
$$\hat{c} = \frac{\int c\, Des_B(s,c)\, dc}{\int Des_B(s,c)\, dc}. \tag{3}$$

The use of (3) has been found satisfactory in our experiments, provided that the rules in a control schema do not suggest dramatically opposite actions in the same situation: when this happens, averaging using (3) simply does not make sense. (The best trade-off between avoiding an incoming train from the left or from the right is seldom to stay on the rails!) Our empirical strategy has been to make sure that the premises of conflicting control rules are not applicable to the same situation; other authors have preferred to use alternative choice rules (e.g., [Yen and Pfluger, 1992]).

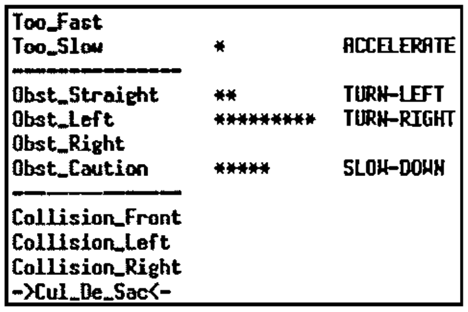
Figure 4 exemplifies Flakey's reactivity. The environment shown is similar to the one at the 1992 robotic competition of AAAI [Congdon *et al.*, 1993a]. Flakey did not have any map of the environment, but simply roamed around at random while trying to stay away from the obstacles. The simple WANDER behavior used in this run consists of 10 rules, shown on the left:[3] from the top, there are two rules for keeping a constant cruising speed, four to smoothly modify the heading and speed of Flakey to keep it far from obstacles, and four rules for engaging in emergency maneuvers when an immediate danger is detected. Each line lists the name of a rule, the truth value of its antecedent in the configuation shown (indicated by the number of stars), and its consequent. The rules used by Flakey in this example are listed in the Appendix.

[3]This behavior is actually composed of three distinct behaviors, and three rule sets — we'll return to this example in Section 5.

(a)



(b)



```
Too_Fast
Too_Slow          *            ACCELERATE
---------------
Obst_Straight     **           TURN-LEFT
Obst_Left         *********    TURN-RIGHT
Obst_Right
Obst_Caution      *****        SLOW-DOWN
---------------
Collision_Front
Collision_Left
Collision_Right
->Cul_De_Sac<-
```

Figure 4: A sample run of the WANDER behavior (a), and the corresponding control rules (b).

4 Beyond pure reactivity

The behaviors discussed in the previous section are purely reactive: at each cycle, Flakey selects an action solely on the basis of the current state of the world as perceived by its sensors and represented in the local perceptual space. Purely reactive behaviors, intended to provide quick, simple reactions to potential dangers (e.g., avoiding collisions) typically use sensor data that has undergone little or no interpretation. Engaging into more purposeful activities than just wandering around requires more than pure reactivity: we need to take explicit goals into consideration. For example, we may want Flakey to reach a given position at a given velocity, and still (reactively) avoid the obstacles on the way.[4]

In our approach, a goal is represented by a *control structure.* Intuitively, a control structure is a virtual object (an *artifact*) that we put in the LPS, associated with a behavior that encodes the way to react to the presence of this object. For example, a "control-point" is a marker for a (x, y) location, together with a heading and a velocity: the associated behavior GO-TO-CP reacts to the presence of a control point in the LPS by generating the commands to reach that position, heading and velocity. The circle near the upper left corner of Figure 1 represents a control point. More precisely, a control structure is a triple

$$S = \langle A, B, C \rangle, \tag{4}$$

where A is an artifact, B is a behavior that refers to the artifact, and C is a context of applicability (see below). Such a control structure implicitly defines a goal: the goal to achieve the situation promoted by behavior B with respect to the artifact A. A theoretical analysis of the relationship between control structures and goals is given in [Saffiotti *et al.*, 1993a].

Purposeful behaviors react to the presence of control structures in the LPS by generating a corresponding preference for controls. For example, a behavior for following a wall reacts to the presence of a control structure

$$S1 = \langle \mathit{Wall1}, \textsc{Follow}, \mathit{clear\text{-}path} \rangle$$

[4]Reactive behaviors are also associated with (innate) goals, hardwired in the definition of the behavior. We are now interested in dynamically assigning explicit strategic goals to Flakey.

in the LPS by generating preferences for the commands that keep Flakey parallel to the wall *Wall1* and at a fixed distance and proceeding at a given cruising speed. Hence, putting a control structure in the LPS is the basic way to communicate a goal to the fuzzy controller (provided that the controller includes a corresponding behavior).

Purposeful behaviors are implemented in the same form as reactive behaviors (see Figure 2 above), the only (formally invisible) difference being that the fuzzy state depends in general on properties of the artifact of the control structure — for example, its position relative to Flakey. Correspondingly, the antecedents A_i's of fuzzy rules associated with those behaviors include fuzzy predicates that depend on those properties. For example, the FOLLOW behavior responds to the presence of the control structure $S1$ above by preferring controls that lead Flakey along the wall within a given distance; it includes the following rule:

```
IF too-close-on-right(wall) THEN turn moderate-left
```

where *wall* is the artifact of the control structure to which the behavior is reacting (in this case, *Wall1*).

Our fuzzy controller includes several purposeful behaviors, which react to different types of control structures, for example, FOLLOW for walls and corridor, REACH for locations and control points, CROSS for doors and junctions, and FACE for locations and objects. The behaviors that we are currently using for Flakey typically consist of two to ten rules, using four to twenty fuzzy predicates. The reader can find in the Appendix the actual rules we wrote to implement several of these behaviors.

Figure 5 shows Flakey's LPS window while Flakey was executing a GO-TO-CP. The local perceptual space of Flakey (in top view) may be seen on the right. Flakey appears in the middle of the window, pointing upwards; the small points around on the right and near the top of the window mark sonar readings, indicating the possible presence of some object; the segment marked by a "W" indicates an interpretation of these data: in this case, a wall hypothesis has been generated. The target control point is in front of Flakey: the "tail" indicates the desired entry direction. The movement of flakey is evident in the wake of small rectangles it leaves behind.

The artifacts used in control structures are identifiers of internal variables repre-

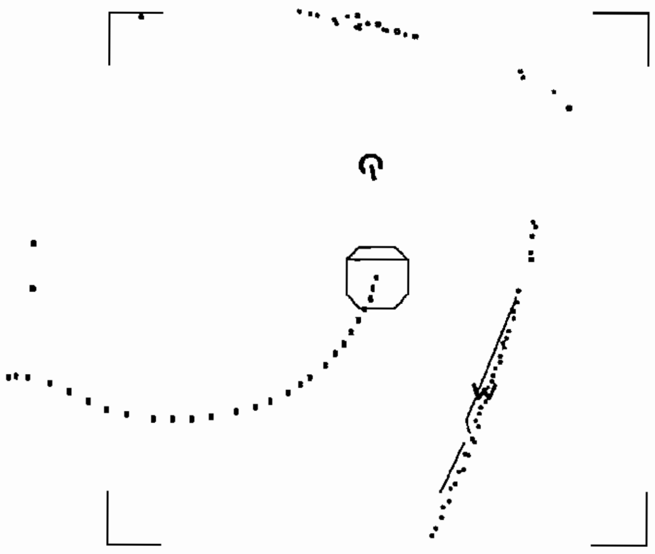


Figure 5: Flakey's LPS while engaged in a GO-TO-CP behavior.

senting the aspects of the state that are relevant to those control structure. Artifacts do not need to correspond to physical objects in the environment: an imaginary spot to reach can be an artifact. When the artifact corresponds to a physical element of the world, however, it provides a way to link, or *anchor* the action of the agent to the environment. To make all this more concrete, consider the following control structure

$$S2 = \langle \mathit{Corr1}, \textsc{Follow}, \mathit{clear\text{-}path} \rangle$$

aimed to move down a given corridor *Corr1*. Figure 6 shows Flakey's LPS during the execution of this control structure. The double lines indicate the corridor artifact; FOLLOW keeps the robot traveling between these lines, assuming that they correspond to the real corridor (a). The position of the physical walls can be seen by looking at the sonar readings (the small dots); because of errors in the map and in Flakey's self-localization, the artifact does not correspond exactly to the real corridor. In (b), however, enough sonar readings have been collected to interpolate two long segments interpreted as the corridor's walls indicated by "W"). The position of the corridor is inferred from such interpretations (indicated by "C"). This position

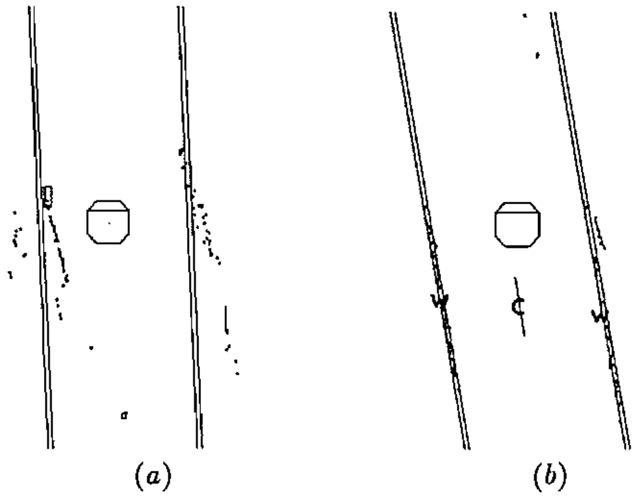


Figure 6: Anchoring the artifact used for action with perception: corridor following before anchoring (a) and after anchoring (b).

is then used to anchor the artifact to the perceived corridor: from now on, the action will take place with respect to the actually perceived corridor. The anchoring mechanism, together with the intrinsic tolerance of fuzzy rules, is essential to allow Flakey to operate by relying only on approximate maps and imprecise sensing.

5 Blending of behaviors

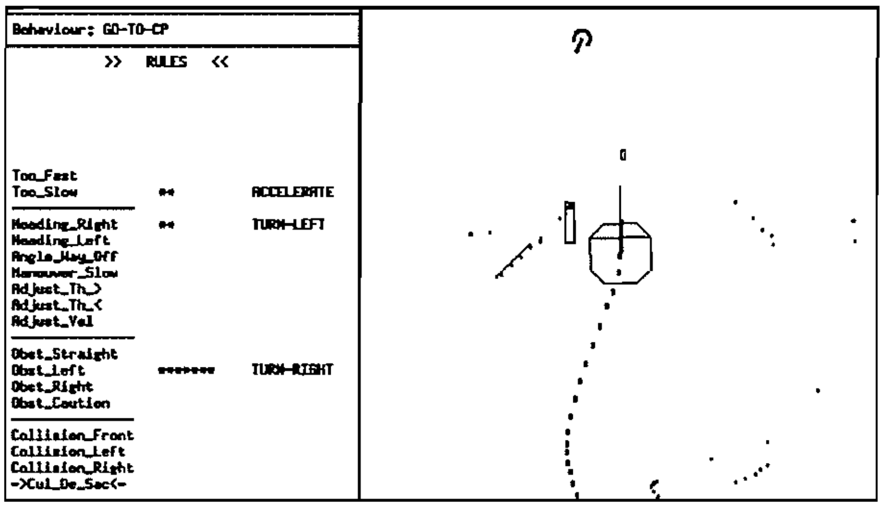
We have seen how our fuzzy controller can implement both reactive and goal-oriented behaviors. As noted in the introduction, several behaviors of both types can coexist in an agent's normal activity: for example, a purposeful behavior for following a wall can coexist with one for keeping the load well balanced, and with another one for avoiding the obstacles along the way. Correspondingly, several instances of the fuzzy machine in Figure 2 are simultaneously active in the fuzzy controller. At any given moment, the robot's overall behavior is the result of the integration, or *blending*, of all the behaviors that are active at that moment.

In its simplest form, blending of behaviors is obtained by combining the corresponding desirability functions into a composite one by means of the $\oplus$ t-conorm (min, in our case):

$$Des(s,c) = Des_1(s,c) \oplus Des_2(s,c) \oplus \ldots \oplus Des_n(s,c)\,,$$

where Des_i is the desirability function produced by the i-th behavior, and then choosing a most desired control for execution using centroid defuzzification (3). This way of composing behaviors is schematized in the architecture shown in Figure 1. Care must be taken, however, of possible conflicts among behaviors aiming at different, incompatible goals. These conflicts would result in desirability functions that assign high values to opposite actions: simple t-conorm composition should not be used in these cases.

The key observation here is that each behavior has in general its own *context* of applicability. Correspondingly, we would like the impact of the control actions suggested by each behavior to be weighted according to that behavior's degree of applicability to the current situation. For example, the FOLLOW-WALL behavior is most applicable when the wall is near and the path is clear; while the preferences expressed by the KEEP-OFF behavior become more relevant when there is an obstacle on the way. To make this precise, recall that the definition (4) of a control structure S_i included a fuzzy predicate C_i expressing its context of applicability. Hence, given n control structures $\{S_1, \ldots, S_n\}$, with corresponding behaviors $\{B_1, \ldots, B_n\}$ and contexts $\{C_1, \ldots, C_n\}$, we define the *context-dependent blending* of these control



```
Behaviour: GO-TO-CP
             >>  RULES  <<

Too_Fast
Too_Slow           **         ACCELERATE
---------------
Heading_Right      **         TURN-LEFT
Heading_Left
Angle_Way_Off
Manouver_Slow
Adjust_Th_>
Adjust_Th_<
Adjust_Vel
---------------
Obst_Straight
Obst_Left          *******    TURN-RIGHT
Obst_Right
Obst_Caution
---------------
Collision_Front
Collision_Left
Collision_Right
->Cul_De_Sac<-
```

Figure 7: A snapshot of Flakey's control window while approaching a control point.

structures to be the behavior characterized by the following desirability function:

$$Des(s,c) = (Des_1(s,c) \circledast C_1(s)) \oplus \ldots \oplus (Des_n(s,c) \circledast C_n(s)) . \tag{5}$$

In practice, context-dependent blending of behaviors is implemented by discounting the output of each behavior using **context rules** of the form

$$\text{IF } C_i \text{ THEN } apply(B_i) .$$

The truth value of the context C_i is used as the **activation level** input to the fuzzy machine in Figure 2. This value is typically a combination of values in the fuzzy state. The output of each fuzzy machine is discounted by the corresponding activation level through $\circledast$ and then combined through $\oplus$ into the overall desirability function as in (5). This function is then defuzzified by (3) to produce a trade-off control, as shown in Figure 1. It is important to notice that the choice of a preferred control is done *after* all the desirability functions have been combined; this contrasts with other approaches to behavior composition where first one preferred control is computed for each behavior, and then all the preferred controls are combined together (e.g., [Arkin, 1990]).

Figure 7 illustrates the combination process as seen in Flakey's control window during an actual run. Flakey's LPS window is on the right; the rectangle on the left of Flakey in this window highlights a dangerously close object. The window on the left lists all the currently active rules, grouped into three rule sets corresponding to the three currently active behaviors: on top, the rules for the GO-FORWARD behavior; below, those for GO-TO-CP, for KEEP-OFF, and for AVOID-COLLISIONS. The context rules for blending these three behaviors, which are not shown in the figure, are

```
IF collision-danger
THEN APPLY(Avoid-Collisions)

IF obstacle-approaching
THEN APPLY(Keep-Off)

IF  NOT(collision-danger)
AND NOT(obstacle-approaching)
THEN APPLY (Go-To-CP, Go-Forward)
```

In the situation depicted in Figure 7, Flakey is going too slowly and heading toward a point to the right of the CP. Some desirability is then assigned to the *accelerate* and the *turn-left* actions. However, the close obstacle on the left causes the activation level of the KEEP-OFF obstacle avoidance behavior to be high, at the expense of the other behaviors. The *turn-right* action associated with KEEP-OFF receives, therefore, the highest possible total desirability (as indicated by the Seven Stars). The small box in front of Flakey indicates the resulting turning control—a few degrees to the right. The overall result of the blending process leads Flakey around obstacles while *en route* to attaining the position and bearing of the control point. Flakey's speed in this experiment was between 200 and 300 mm/s.

As a second example of blending reactive and goal-oriented behavior, consider the task of following a corridor while avoiding possible obstacles. The following context rules express the desired interaction between the obstacle-avoidance behavior KEEP-OFF, and the corridor-following behavior FOLLOW.

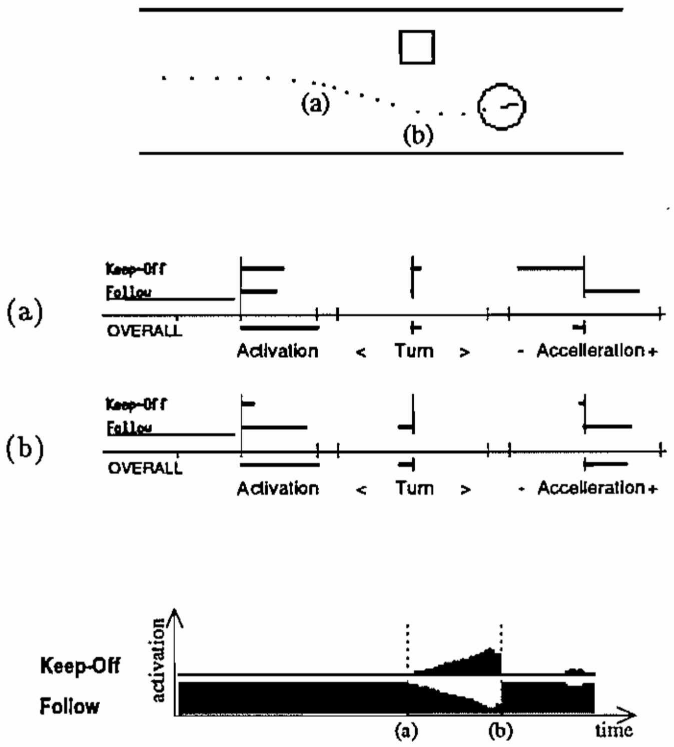


Figure 8: Another example of blending reactive and purposeful behavior.

```
IF collision-danger
THEN APPLY(Keep-Off)

IF  NOT(collision-danger)
THEN APPLY(Follow)
```

Figure 8 shows the result of this blending. We have chosen here a more concise graphical representation: the bars graphs show the activation level and the preferred controls (turn and acceleration) for each behavior at two critical instants; the bottom line in each graph shows the result of the blending. In (a), an obstacle has been detected, and the preferences of KEEP-OFF are dominating; later, when the path is clear, the goal-oriented preferences expressed by FOLLOW regain importance (b). The graph at the bottom of Figure 8 plots the activation level of each behavior over time.

It is interesting to go back to the wandering example considered in Section 3 — see Figure 4. The WANDER behavior is actually obtained by blending three distinct behaviors: KEEP-OFF, AVOID-COLLISIONS (a behavior that looks at the nearest sonar readings and proposes drastic actions when an immediate risk of collision is detected), and GO-FORWARD (which keeps Flakey going at a fixed cruising velocity, given as a parameter, when there are no obstacles in the way). The activation levels of KEEP-OFF and AVOID-COLLISIONS are given by the state variable "approaching-obstacle"; the complement of this value gives the activation level for GO-FORWARD. The visual result for an external observer is that Flakey "follows its nose", while smoothly turning away from obstacles as it approaches them.

The examples above considered the blending of a single goal-achieving behavior with reactive behavior. In general, many goals can be considered simultaneously; in particular, all the (sub)goals that form a full plan can be represented as control structures, each one associated with a context that describes the situations where a particular goal becomes relevant. In this regard, control structures act as an adequate declarative representation for action that can be shared between a classical symbolic planner and the fuzzy controller. In Figure 1, the planner generates plans in the form of sets of control structures, and puts these control structures in the LPS to be considered (modulo their context) by the fuzzy controller. In this way, our approach can combine high-level symbolic planning techniques with low-level continuous numerical control, without requiring substantial conceptual modifications to the traditional roles of planners and controllers in AI approaches. The companion technical report [Saffiotti *et al.*, 1993a] elaborates more on this form of integration between classical AI planning and fuzzy control.

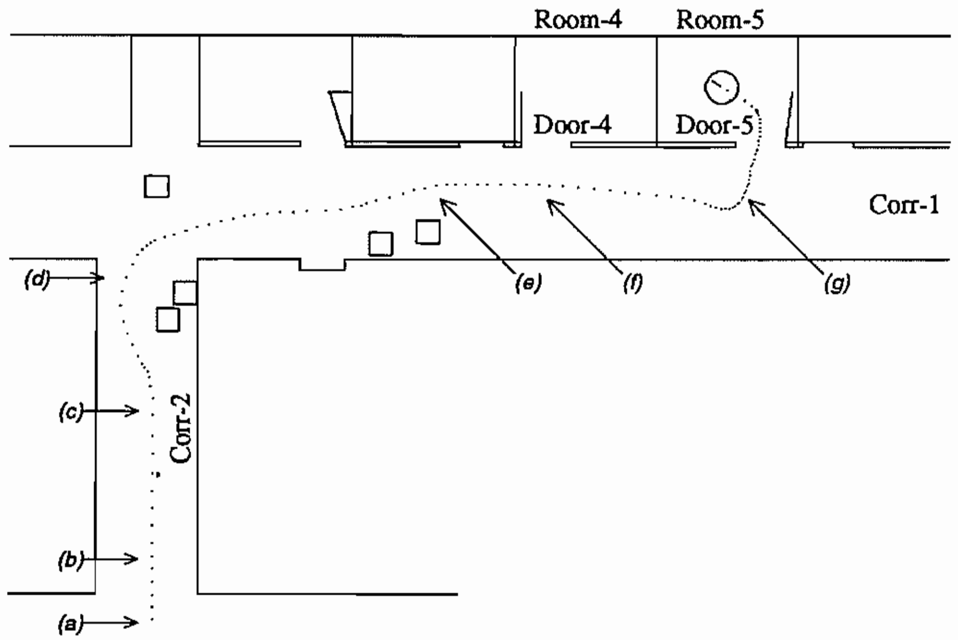
To better understand the use of sets of control structures as complex plans, consider the example shown in Figure 9. Flakey had been given the goal to visit Room-5, and used a simple goal regression planner to generate a corresponding set of control structures based on a sparse topological map annotated with approximate metric information. No obstacle was represented in the map. The plan, in a simplified form, consists of the following four control structures[5]

$$\begin{aligned} S1 &= \langle \text{Obstacle}, \textsc{Keep-Off}, \mathit{near(Obstacle)} \rangle \\ S2 &= \langle \text{Corr2}, \textsc{Follow}, \neg \mathit{near(Obstacle)} \wedge \mathit{at(Corr2)} \wedge \neg \mathit{near(Corr1)} \rangle \\ S3 &= \langle \text{Corr1}, \textsc{Follow}, \neg \mathit{near(Obstacle)} \wedge \mathit{at(Corr1)} \wedge \neg \mathit{near(Door5)} \rangle \\ S4 &= \langle \text{Door1}, \textsc{Cross}, \neg \mathit{near(Obstacle)} \wedge \mathit{near(Door5)} \rangle \end{aligned}$$

Figure 9 shows an instance of an execution of this plan, along with the corresponding level of activation over time of each control structure in the actual plan. Total execution time was approximately 80 s, at top speeds of 400 mm/s.

The dynamics of the contexts results in sequencing the execution of the control structures: at (a), Flakey is *at(Corr2)* and $S2$ is most active; at (d), Flakey transitions from being *at(Corr2)* to being *at(Corr1)*, and the activation level of $S3$ begins to increase, at the expense of $S2$. A similar pattern of smooth sequencing occurs around (g). Notice also the interaction between purposeful behaviors and reactive obstacle avoidance — after (c), and around (d), (e), and (g). In particular, the interaction right after (g) is worth a comment. The Cross behavior relies on prior information about the door position; as this turns out to be fairly off, Keep-Off raises to avoid a collision with the edge of the wall. The combined effect is that Flakey engages the opening that is "more or less at the estimated position." The ability to integrate the map knowledge available at planning time with the perceptual knowledge available at execution time is one advantage of using control structures as a representation of a plan shared between the symbolic level of the planner, and the perceptuo-motor level of the fuzzy controller.

[5]The actual plan has more control structures, including some for perceptual actions—the Sense control structures shown in the picture—and more complex contexts; see [Saffiotti *et al.*, 1993a] for a fuller treatment of this example.



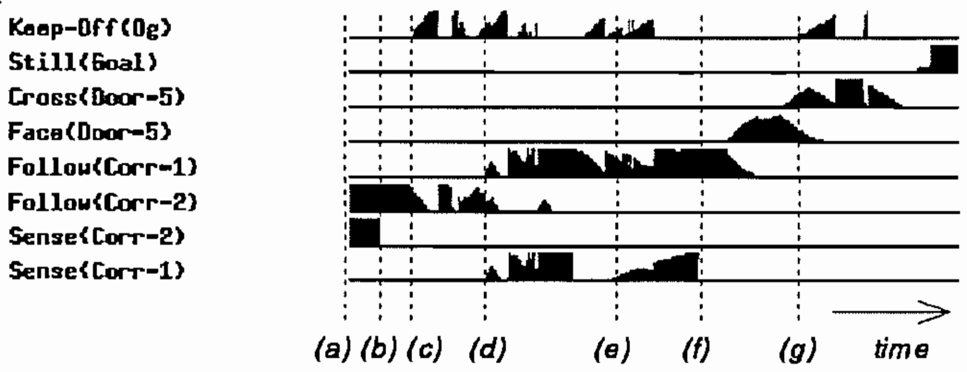


Figure 9: Sample execution of a full plan. The temporal evolution of the activation levels of the control structures in the plan is shown in the lower part.

6 Conclusions

We have defined a mechanism based on fuzzy logic for blending multiple behaviors aimed at achieving different, possibly conflicting goals, and discussed its implementation on the SRI mobile robot, Flakey. Goals are either built-in, as in most fuzzy controllers, or dynamically set from outside the controller. Typically, the built-in goals correspond to reactive behaviors (like avoiding collisions), while the dynamic ones are motion or perceptual strategic goals communicated by a planner. Context-dependent blending of behaviors ensures that strategic goals be achieved as much as possible, while a high reactivity is maintained.

Our behavior blending mechanism has been originally inspired to the technique proposed by Berenji et al. [Berenji *et al.*, 1990] for dealing with multiple goals in fuzzy control. There are, however, two important differences: first, our context mechanism dynamically modifies the degrees of importance of each goal; second, we allow the introduction of high-level, situation-specific goals in the controller.

From another perspective, the work presented here fits in the tradition of the "two-level" approaches to robot control, where a strategic planner is used to generate guidelines to a reactive controller (e.g., [Arkin, 1990; Payton *et al.*, 1990; Gat, 1991]). In our case, a plan consists in a set of control structures. Hence, our plans can find close relatives in some of the recent approaches to autonomous agency where plans are seen as sets of *condition* $\rightarrow$ *action* rules [Suchman, 1987; Schoppers, 1987; Payton *et al.*, 1990]. However, we believe that basing our architecture on multi-valued logics results in improved robustness (e.g., more tolerance to sensor noise and knowledge imprecision), while granting a better understanding of the underlying combination mechanisms.

Finally, many current approaches to robot control deal with multiple goals using the so-called "potential fields" method [Khatib, 1986; Arkin, 1990]: goals are represented by pseudo-forces, which may be thought of as representatives of most desirable behavior from that goal's viewpoint. These optimal forces are then combined, as physical vectors, to produce a resultant force that summarizes their joint effect. In our approach, by contrast, the goals' desirability functions, rather than a summary description, are combined into a joint desirability function, from which a most desired trade-off control is extracted. We have already noticed that desirability

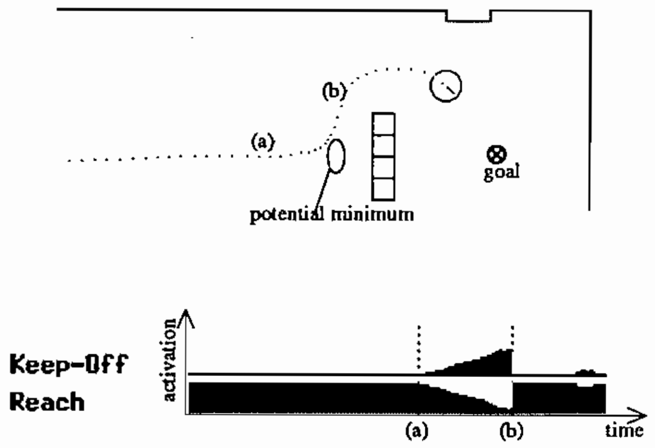


Figure 10: How context-dependent blending of behaviors helps to avoid potential local minima.

functions are less committal than classical control functions—or pseudo-potentials—in that they map each input state to a fuzzy set of possible controls rather than to one "best" control. One technical advantage of describing control by multivalued desirability mappings is that suboptimal combinations can be more precisely characterized [Saffiotti *et al.*, 1993a].

A second difference between our behavior combination technique and most potential-field based methods is that our approach takes behaviors' context of applicability into account; this provides greater control over the combination, and helps in managing the local minima possibly arising from the local combination of behaviors [Latombe, 1991]. The problem is illustrated in Figure 10 (top): the robot needs to mediate the tendency to move toward the goal, and the tendency to stay away from the obstacle. A straightforward combination of these two opposite tendencies (whether they are described by desirability measures, potential fields [Khatib, 1986], motor schemas [Arkin, 1990], or formalisms) may result in the production of a zone of local equilibrium (local minimum): when coming from the left edge, the robot would be first attracted and then trapped into this zone. By using context rules to reason about the relative importance of goals, context-dependent blending of behaviors provides a way around this problem. Figure 10 shows the path followed by Flakey in a simulated run (top), and the corresponding activation levels of the KEEP-OFF and REACH behaviors (bottom). In (a), Flakey has perceived the obstacle; as the obstacle becomes nearer, the KEEP-OFF behavior becomes more active, at the expenses of the REACH behavior. In this way, the "attractive power" of the goal is gradually shaded away by the obstacle, and Flakey responds more and more to the obstacle-avoidance suggestions alone. The REACH behavior regains importance, however, as soon as Flakey is out of danger (b).

As a final difference between our technique and potential-field methods, we mention that we express both goals and applicability conditions as formulae in a (multi-valued) logical language. Complex goals and constraints can often be described more easily in a logical form than in the analytical form of a potential field function. Moreover, the ability to specify the context used in the combination as a logical proposition provides a hook to higher-level symbolic processes: we have seen above that a planner can produce complex controls by associating the right context to each behavior.

The technique proposed in this paper has been implemented in the SRI mobile robot Flakey, resulting in extremely smooth and reliable movement in navigating in unstructured, real-world environments. The performance of Flakey's controller has been demonstrated on various occasions, including innumerable runs in the corridors and offices of the AI Center during working hours, and pamphlet distribution during crowded coffee-breaks at the second IEEE Conference on Fuzzy Sets and Systems (San Francisco, CA, April 1993). Flakey's ability was also demonstrated at the first AAAI robotic competition, held in San Jose, CA, in July 1992 [Congdon *et al.*, 1993b]. Flakey accomplished all the given tasks while smoothly getting around obstacles (whose positions were not known beforehand) and people, and placed overall second. Flakey's reliable reactivity is best summarized in one judge's comment: "Only robot I felt I could sit or lie down in front of." (What he actually did!)

Acknowledgments John Lowrance, Daniela Musto, Karen Myers, and Leonard Wesley contributed to the development of the ideas presented in this paper. Nicolas Helft implemented a first version of Flakey's controller.

References

[Arkin, 1990] R. C. Arkin. The impact of cybernetics on the design of a mobile robot system: a case study. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(6):1245–1257, 1990.

[Berenji *et al.*, 1990] H. Berenji, Y-Y. Chen, C-C. Lee, J-S. Jang, and S. Murugesan. A hierarchical approach to designing approximate reasoning-based controllers for dynamic physical systems. In *Procs. of the 6th Conf. on Uncertainty in Artificial Intelligence*, Cambridge, MA, 1990.

[Congdon *et al.*, 1993a] C. Congdon, M. Huber, D. Kortenkamp, C. Bidlack, C. Cohen, S. Huffman, F. Koss, U. Raschke, T. Weymuth, K. Konolige, K. Myers, A. Saffiotti, E. H. Ruspini, and D. Musto. CARMEL vs. Flakey: A comparison of two winners. Technical report series, AAAI, Menlo Park, California, 1993. Short version appeared in *AI Magazine*, 14(1), Spring 1993, pp. 49–57.

[Congdon *et al.*, 1993b] C. Congdon, M. Huber, D. Kortenkamp, K. Konolige, K. Myers, E. H. Ruspini, and A. Saffiotti. CARMEL vs. Flakey: A comparison of two winners. *AI Magazine*, 14(1):49–57, Spring 1993.

[Firby, 1987] J. R. Firby. An investigation into reactive planning in complex domains. In *Procs. of the AAAI Conf.*, 1987.

[Gat, 1991] E. Gat. *Reliable Goal-Directed Reactive Control for Real-World Autonomous Mobile Robots.* PhD thesis, Virginia Polytechnic Institute and State University, 1991.

[Kaelbling, 1987] L. P. Kaelbling. An architecture for intelligent reactive systems. In M.P. Georgeff and A.L. Lansky, editors, *Reasoning about Actions and Plans.* Morgan Kaufmann, 1987.

[Khatib, 1986] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[Latombe, 1991] J.C. Latombe. *Robot Motion Planning.* Kluver Academic Publishers, Boston, MA, 1991.

[Payton *et al.*, 1990] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey. Plan guided reaction. *IEEE Trans. on Systems, Man, and Cybernetics*, 20(6), 1990.

[Ruspini and Saffiotti, 1993] E. H. Ruspini and A. Saffiotti. The formulation and solution of non-linear optimization problems in multi-valued logic. Technical report, SRI Artificial Intelligence Center, Menlo Park, California, 1993. Forthcoming.

[Ruspini, 1990] E. H. Ruspini. Fuzzy logic in the Flakey robot. In *Procs. of the Int. Conf. on Fuzzy Logic and Neural Networks (IIZUKA)*, pages 767–770, Japan, 1990.

[Ruspini, 1991a] E. H. Ruspini. On the semantics of fuzzy logic. *Int. J. of Approximate Reasoning*, 5:45–88, 1991.

[Ruspini, 1991b] E. H. Ruspini. Truth as utility: A conceptual systesis. In *Procs. of the 7th Conf. on Uncertainty in Artificial Intelligence*, Los Angeles, CA, 1991.

[Saffiotti *et al.*, 1993a] A. Saffiotti, K. Konolige, and E. H. Ruspini. A multivalued-logic approach to integrating planning and control — or, now, do it! Technical Report 533, SRI Artificial Intelligence Center, Menlo Park, California, 1993.

[Saffiotti *et al.*, 1993b] A. Saffiotti, E. H. Ruspini, and K. Konolige. A fuzzy controller for flakey, an autonomous mobile robot. In *Procs. of the Fuzzy Logic '93 Conference*, Burlingame, CA, 1993.

[Saffiotti *et al.*, 1993c] A. Saffiotti, E. H. Ruspini, and K. Konolige. Integrating reactivity and goal-directedness in a fuzzy controller. In *Procs. of the 2nd Fuzzy-IEEE Conference*, San Francisco, CA, 1993.

[Saffiotti, 1993] A. Saffiotti. Some notes on the integration of planning and reactivity in autonomous mobile robots. In *Procs. of the AAAI Spring Symposium on Foundations of Automatic Planning*, pages 122–126, Stanford, CA, 1993.

[Schoppers, 1987] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Procs. of the Int. Joint Conf. on Artificial Intelligence*, pages 1039–1046, 1987.

[Schweizer and Sklar, 1983] B. Schweizer and A. Sklar. *Probabilistic metric spaces.* North-Holland, Amsterdam, NL, 1983.

[Suchman, 1987] Lucy Suchman. *Plans and situated actions: the problem of human machine communication.* Cambridge University Press, Cambridge, MA, 1987.

[Sugeno and Nishida, 1985] M. Sugeno and M. Nishida. Fuzzy control of a model car. *Fuzzy Sets and Systems*, 16:103–113, 1985.

[Yen and Pfluger, 1992] J. Yen and N. Pfluger. A fuzzy logic based robot navigation system. In *Procs. of the AAAI Fall Symposium on Mobile Robot Navigation*, pages 195–199, Boston, MA, 1992.

[Zadeh, 1978] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

Appendix: fuzzy rules for some of Flakey's behaviors

This appendix lists the fuzzy rules for the main behaviors implemented for Flakey. The rules are as they appear in the Lisp code of Flakey's controller, uncommented, and are only shown to give the reader a feeling of the use of fuzzy rules as a robot programming language. For the most part, however, their meaning should be reasonably intuitive. All the variables used in the antecedents are in the *fuzzy state* of the corresponding behavior, and take values in $[0, 1]$; appropriate update functions, not shown, set these variables at every cycle based on the current content of the Local Perceptual Space.

```
;;; --------------- Avoid Collisions ---------------
;;
;; Emergency maneuvers when an immediate danger is detected
;;

(defrule :->cul_de_sac<-
         :antecedent ((and~ (not-moving) (dans-un-cul-de-sac)
                            (not~ (escaping-from-cul-de-sac))))
         :consequent :turn-left)

(defrule :collision_right
         :antecedent ((and~ (or~ collision-right collision-front-right)
                            (not~ collision-front-left)))
         :consequent :turn-left)

(defrule :collision_left
         :antecedent ((and~ (or~ collision-left collision-front-left)
                            (not~ collision-right)))
         :consequent :turn-right)

(defrule :collision_front
         :antecedent (collision-front)
         :consequent :slow-down)
```

```
;;; --------------- Keep Off ---------------
;;
;; Stay away from occupied areas
;;

(defrule :obst_caution
         :antecedent ((and~ too-fast-for-obstacle
                            (or~ obstacle-in-front
                                 (and~ obstacle-left-side obstacle-right-side))))
         :consequent :slow-down)

(defrule :obst_right
         :antecedent ((or~ obstacle-right-side
                           (and~ obstacle-angled-right
                                 obstacle-near
                                 (not~ obstacle-left-side))))
         :consequent :turn-left)

(defrule :obst_left
         :antecedent ((or~ obstacle-left-side
                           (and~ obstacle-angled-left
                                 obstacle-near
                                 (not~ obstacle-right-side))))
         :consequent :turn-right)

(defrule :obst_straight
         :antecedent ((and~ obstacle-straight
                            (not~ obstacle-left-side)
                            (not~ obstacle-right-side)))
         :consequent *preferred-obstacle-turn*)


;;; --------------- Keep Velocity ---------------
;;
;; Maintain a constant cruising velocity
;;

(defrule :too_slow
         :antecedent (too-slow)
         :consequent :accelerate)

(defrule :too_fast
         :antecedent (too-fast)
         :consequent :slow-down)
```

```
;;; --------------- Go To Position ---------------
;;
;; Reach a (X, Y) position, and stop
;;

(defrule :pos_on_left
         :antecedent ((and~ position-on-left (not~ position-achieved)))
         :consequent :turn-left)

(defrule :pos_on_right
         :antecedent ((and~ position-on-right (not~ position-achieved)))
         :consequent :turn-right)

(defrule :pos_here!
         :antecedent ((and~ position-achieved moving))
         :consequent :slow-down)



;;; --------------- Follow ---------------
;;
;; Run within a given (virtual) lane
;; e.g., to follow a wall, or a corridor
;;

(defrule :angled_left
         :antecedent ((and~ rhs-clear angled-left near-lane))
         :consequent :turn-right)

(defrule :angled_right
         :antecedent ((and~ lhs-clear angled-right near-lane))
         :consequent :turn-left)

(defrule :left_of_lane
         :antecedent ((and~ rhs-clear on-left (not~ angled-right)))
         :consequent :turn-right)

(defrule :right_of_lane
         :antecedent ((and~ lhs-clear on-right (not~ angled-left)))
         :consequent :turn-left)
```

```
;;; --------------- Go To Control Point ---------------
;;
;; Achieve a given (X, Y, THETA, VEL) state
;;

(defrule :adjust_vel
         :antecedent ((and~ near-cp (not~ velocity-ok)))
         :consequent :accelerate)

(defrule :adjust_th_<
         :antecedent ((and~ near-cp cp-on-left))
         :consequent :turn-left)

(defrule :adjust_th_>
         :antecedent ((and~ near-cp cp-on-right))
         :consequent :turn-right)

(defrule :manouver_slow
         :antecedent ((and~ around-cp (not~ heading-ok)))
         :consequent :slow-down)

(defrule :heading_left
         :antecedent ((and~ heading-left rhs-clear (not~ near-cp)))
         :consequent :turn-right)

(defrule :heading_right
         :antecedent ((and~ heading-right lhs-clear (not~ near-cp)))
         :consequent :turn-left)
```