Finding Referents While Following Directions

Andrew R. Haas
Department of Computer Science
University at Albany, State University of New York
1400 Washington Avenue
Albany, New York 12222
(518) 442-9066
haas@cs.albany.edu

Introduction

If a robot accepts directions in natural language, it must find the referents of NP's in the directions. A robot cannot always find the referents before it starts to execute the directions. It is often necessary to execute part of a command, find a referent, and continue execution. Here is an example, drawn from an experiment in which subjects give directions to a simulated robot. The robot is standing at the intersection of two corridors; there is no door in sight. The subject types the following:

1 Turn right and go through the door.

The robot turns right and sees a single door in plain sight. It should identify this door as the referent of "the door", and then plan to go through the door. The robot must turn right *before* it identifies the referent of "the door". Therefore it must plan and execute the right turn before it can identify the argument of the next action.

Along with NP referents, the robot must identify the starting and finishing times of actions. These are described by adverbials. We get examples like

2 Go down the hall and stop at the table.

To understand this example, we must use the distinction between an *activity* and an *accomplishment* (Dowty 1979). An accomplishment is finished when the agent reaches a certain state. For example, the clause "Go through the door" describes an accomplishment. When you reach the other side of the doorway, the accomplishment is finished. There is no need for the speaker to specify the finishing time. To complete an activity, the agent does not have to reach any particular state. The clause "Go down the hall" describes an activity. Since there is no state that marks the finishing of this activity, and there is no adverbial describing the finishing time, this clause leaves the hearer in doubt about when to stop going down the hall.

In example (2), it is clear that the robot should stop going down the hall when it reaches the table. Why is this? The adverbial "at the table" does not modify the verb "go" – it modifies the next verb. Command (2) is an example of a pattern that is common in directions. When an activity occurs in directions, followed by another action, a hearer will continue the activity until he is ready to begin the next action. We say "Go north on I-87 and take exit 5". The hearer understands that he should go north on I-87 until he sees exit 5. That is: he should continue the activity "Go north on I-87" until he is ready to start the next action. In the same way, the hearer of (2) will continue the activity "go down the hall" until he is ready to start the next action. The adverbial "at the table" indicates that he will be ready to start the next action when he reaches the table.

This paper describes a simulated robot that can follow simple directions. 3-D graphics allow the user to watch the robot's actions. I have collected a small number of dialogues, in which subjects direct the robot through a building to a given destination. The examples in this paper are drawn from this experiment.

The robot does not plan to search for referents. It assumes that as it follows the directions, the referents will appear when they are needed. The problem is to determine where in the plan a referent is needed. The robot solves this problem by taking a semantic representation from its parser, and running it through a pre-processor. This program re-arranges the semantic representation, attaching the descriptions of referents to the points in the plan where those referents are needed. This produces a high-level plan, which is executed by a second program. Section 2 of this paper describes the semantic representations; section 3 describes the pre-processor; section 4 describes the execution of the high-level plans.

Semantic Representation

In example (1), "the door" refers to a door that is salient to the utterance (because it is in plain sight). To represent this salience in formal semantics, many authors use a *context* – a vector of properties of an utterance that are needed to determine its semantics. This vector might include speaker, hearer, time, place, and a set of salient entities (Lewis 1972).

My program relies on a different approach to formal semantics. There is no vector, containing all properties of the utterance that are needed for semantics. Each semantic rule refers to whatever properties of the utter-

ance it needs. Let U be an utterance of a sentence R. The logical form of R, together with certain properties of the utterance U, will determine the set S of singular propositions that U can express (Kaplan 1989). I suggest that the logical form of R is a pair (P, Q), where P and Q are formulas with free variables. The free variables of P represent the referents of indexicals and demonstratives. Q expresses a constraint on the values of these variables. Since the values depend on the utterance U, we use another free variable u to represent the utterance. Let s be an assignment of values to the free variables of Q such that $s(u) = U$ and s satisfies Q. Then the values assigned by s satisfy the constraint Q. Replacing the free variables of P with the values assigned by s, we get a singular proposition which is a possible interpretation of utterance U.

Suppose U_1 is an utterance of "I love you". The speaker is John and the hearer is Mary. The logical form of this sentence is the pair

3 $(\text{love}(x, y), (\text{speaker}(x, u) \wedge \text{hearer}(y, u)))$

The variable x represents the speaker, and y represents the hearer. Consider the assignment s such that $s(u) = U_1$, $s(x)$=John, and $s(y)$=Mary. This assignment satisfies the constraint $(\text{speaker}(x, u) \wedge \text{hearer}(y, u))$. Therefore $(\text{love}(x, y), s)$ is the singular proposition expressed by the utterance u_1.

Let us apply this technique to examples (1) and (2). In (1), "the door" refers to an object that has never been mentioned before. This object is still invisible at the time of the utterance. Then what makes it salient? Apparently it is salient because it becomes visible after the robot has executed the first conjunct of the command ("turn right"). To express this idea, I suggest a salience predicate with two arguments. $\text{salient}(x, t)$ means that object x is salient at time t. The NP "the door" refers to a time after the robot has turned right. At this time the door is salient because it is in plain sight.

I propose the following representation for example (1).

4

$$((\text{turn}(h,\text{right},t_1) \wedge \text{go}(h,d,t_2)),$$
$$(\text{hearer}(u, h) \wedge t_2 > t_1 \wedge \text{door}(d) \wedge \text{salient}(d,\text{end}(t_1)))$$
$$)$$

The variable h represents the hearer of the utterance; t_1 and t_2 are the intervals of time when the hearer executes the actions; d is the referent of "the door". This referent must be a door, and it must be salient at the end of interval t_1 – that is, when the robot has finished turning right. The constraint in (3) is a conjunction of atoms, and all constraints used in this paper will also be conjunctions of atoms.

Example (2) has the following representation:

5

$$(\text{go}(h,\text{down}(x),t_1) \wedge \text{stop}(h, t_2),$$
$$(\text{hearer}(u, h) \wedge \text{time}(u, n) \wedge \text{hall}(x) \wedge \text{salient}(x, n)$$
$$\wedge\ t_1 < t_2 \wedge \text{table}(t) \wedge \text{salient}(t, n)$$
$$\wedge\ \text{at}(h, t,\text{start}(t_2))$$

Again h is the hearer, and t_1 and t_2 are the intervals of time when the hearer executes the actions. n (for "now") is the time of the utterance, x is "the hall", and t is "the table". The hearer should be at the table at the beginning of t_2.

Note that in representation (5), the hall and the table are salient at the time of the utterance. Unlike "the door" in (4), "the table" does not refer to an object that becomes salient after the robot has finished the previous action. The robot's grammar allows a definite description to refer either to the time of the utterance, or to the end of the previous action. The first option is the default; so the robot prefers to interpret a definite description as referring to an object that is salient at the time of the utterance. If there is no object satisfying the description, the robot considers the second option. For example, suppose there is a door in plain sight, and user types in utterance (1). The robot will interpret "the door" as the one in plain sight.

The Pre-Processor

Semantic representations (4) and (5) consist of a series of commands, and a constraint on the values of free variables in these commands. We can divide the arguments of the commands into three groups: the agent, the time of the action, and the remaining arguments, called the *objects* of the command. For example, in representation (4) the object of the second action is the door.

There are two variables whose values can be identified at once: the hearer, and the time of the utterance. The hearer is the robot itself; it replaces the variable that represents the hearer with the constant that it standardly uses to refer to itself. To identify times, the robot relies on an internal clock. When the robot consults the clock, the clock returns a term of the form $\text{time}(i)$, where i is an integer. By definition, this term denotes the time when the robot consulted the clock. I assume that if $i < j$, $\text{time}(i)$ is before $\text{time}(j)$. The numbers i and j do not indicate the length of the interval from $\text{time}(i)$ to $\text{time}(j)$. So the "clock" does not have to measure time; all it needs to do is remember the last integer that it returned. When the sensors produce a description of the robot's surroundings, they use the current reading of the clock to indicate the time when this description is true. When the user types in an utterance, the robot consults the clock to obtain a name for the time when the utterance occurred. It substitutes this name for the variable that represents the time of the utterance. See (Haas 1995) for the use of an internal clock in a robot.

In representations (4) and (5), the constraint $t_1 < t_2$ indicates the order in which the commands are to be executed. The pre-processor uses these order constraints to sort the commands, producing a list of commands in order of execution time. These commands are called steps of the plan.

The given representations do not include variables that represent the start and finish times of the actions. The variables t_1 and t_2 represent the entire interval in which an action is performed. The pre-processor replaces each variable t_i that represents an interval with a pair (s_i, f_i), where s_i is the starting point of the interval, and f_i is the ending point. It then simplifies the representation; for example, it replaces the term end$((s_i, f_i))$ with f_i.

After these preliminary steps, representations (4) and (5) take the following form.

6
([turn(me,right,(s_1, f_1)),go(me,d,(s_2, f_2))],
(door(d) $\wedge$ salient(d, f_1)))
)
7
([go(me,down(x),(s_1, f_1)),stop(me,(s_2, f_2))],
(hall(x) $\wedge$ salient(x,time(500)) $\wedge$ table(t)
$\wedge$ salient(t,time(500)) $\wedge$ at(me,t, s_2))

For each step in the plan, the pre-processor finds the variables that represent the starting time, the finishing time, and the objects of the action. For each of these variables, it chooses a *description*: a subset of the conjuncts in the given constraint that will be used to identify the value of that variable. The description of a variable need not include every conjunct that mentions that variable. In example (6) the formula salient(d, f_1) mentions the finishing time f_1 of the first action. If the robot includes this formula in the description of f_1, it will try to identify "the door" in order to identify the finishing time of the first step in plan (6). It seems better to identify "the door" at the next step.

The pre-processor uses the syntax of the input to decide which conjuncts belong in the description of a variable. The features of each phrase in the parse tree include a list of conjuncts that originate in that phrase. Suppose that clause C_1 describes a step S_1 in a plan, and clause C_2 describes a later step S_2 in the same plan. The descriptions of the starting time, finishing time, and objects of S_1 cannot include any conjunct that originates in clause C_2. The first clause of (1) describes the first step of (6), but the conjunct salient(d, f_1) originates in the second clause of (1). Therefore the description of the finishing time of the first step in (6) cannot include the conjunct salient(d, f_1). This heuristic is far from a complete solution of the problem. However, I believe it is based on a sound principle: that the syntax of commands provides valuable clues for the planner.

Following this heuristic, the pre-processor attaches three descriptions to each action: one to describe the starting time, one to describe the finishing time, and one to describe the objects. It returns the following high-level plans for examples (1) and (2).

8
[step(turn(me,right,(s_1, f_1)),true,true,true),
step(go(me,d,(s_2, f_2)),true,true,
(door(d) $\wedge$ salient(d, f_1)))]

9
[step(go(me,down(x),(s_1, f_1)),true,true,
(hall(x) $\wedge$ salient(x, u,time(500)))),
step(stop(me,(s_2, f_2))),
(at(me,t, s_2) $\wedge$ table(t) $\wedge$ salient(t,time(500))),true,true)]

In the term step(W, P, Q, R), W is an atomic wff that describes an action; P is a description of the starting time of the action, Q a description of its finishing time, and R a description of its objects. In the second step of plan (8), the argument d is a door that is salient at the end of the first step. In the second step of plan (9), the starting time s_2 must satisfy at(me,t, s_2) – that is, the robot must be at the table when it stops.

Plan Execution

The robot considers the steps of a plan in order. Given a step of the form step(W, P, Q, R), the robot first consults its clock and gets a name for the present time. It checks that this time satisfies the starting constraint P. In other words: it checks that now is the time to perform the next action. Normally the time will satisfy the starting constraint, because previous steps have established the starting conditions of the action.

Next the robot proves an instance of the object constraint R. That is, it finds a set of values for the object variables which satisfy the constraint. It substitutes these values into the wff W, which describes the action. Then it calls a low-level planner, which tries to find a plan for executing action W. This planner uses a straightforward route-planning algorithm (Lozano-Perez and Wesley, 1979) and produces a low-level plan whose steps are forward movements and turns. If the planner fails, the robot assumes that it has chosen the wrong values for the object variables; it tries to prove another instance of the object constraint W, and runs the planner again.

If the planner succeeds, the robot executes the low-level plan. It then tries to verify that the action has been successfully completed, by proving an instance of the wff W. The objects in W are now represented by ground terms, but the starting and finishing times are still represented by variables. If the robot proves an instance of W, it has found the values of these variables; it substitutes them into the plan, and continues to the next step.

Consider the execution of plan (8). In the first step, all three descriptions are trivial. The robot builds and executes a plan to turn right. Then it proves an instance of the wff turn(me,right,(s_1, f_1)). That is, it proves that it has turned right. Suppose the robot turned right between time(25) and time(50). It proves the wff turn(me,right,(time(25),time(50))), and substitutes the terms time(25) and time(50) for the variables s_1 and f_1 in the plan. The final step now looks like this:

10
step(go(me,d,(s_2, f_2)),true,true,
(door(d) $\wedge$ salient(d,time(50))))

To execute this step, the robot first proves an instance of the object constraint. That is, it searches for a door that is salient at time 50 (the time when it finished turning right). If there is exactly one door in plain sight at time 50, it substitutes the name of this door for the variable d, and then builds and executes a low-level plan for going through the door.

If the first step in a plan is an activity, the execution is more complex. The robot first chooses values for the object variables, and executes the activity for a short time. Then it decides whether it is ready to begin executing the next action. It consults the clock to find a term denoting the present time, and checks that this time satisfies the starting constraint of the next action. If so, it continues on to the next action. If not, it repeats the first action and checks again.

To execute plan (9), the robot first identifies the object variable x (representing the hall), and then goes a short way down the hall. It consults the clock and finds that the present time is time(100). It substitutes this value for the variable s_2 in the starting constraint of the second action in plan (9). Then it tries to prove an instance of this constraint – that is, it tries to prove that it is now "at the table". This involves identifying "the table". If the robot prove an instance of this constraint, it proceeds to the next action (i.e., it stops). If it fails to prove such an instance, it has not yet reached the table. Therefore it repeats the first action, and checks again.

Conclusions

These algorithms have been successfully tested on 11 sets of directions, formed by editing the directions given by subjects in my experiment. I and my students are now undertaking another set of experiments; we hope to collect about 200 examples, which will serve as data for the next version of the program. The domain raises a wide variety of problems, yet it is limited enough to be tractable. I believe it provides an excellent test for our understanding of the relations between language, knowledge, and action.

Acknowledgements. This work was supported by the National Science Foundation under grant number IRI9402390.

References

Dowty, David. 1979. Word Meaning and Montague Grammar. Dordrecht: D. Reidel.

Haas, Andrew. 1995. Planning to Find the Referents of Noun Phrases. to appear in *Computational Intelligence.*

Kaplan, David. 1989. Demonstratives. in Almog, Joseph; Perry, John; and Wettstein, Howard, eds., Themes from Kaplan, pp. 481-563. New York: Oxford University Press.

Lewis, David. 1972. General Semantics. in D. Davidson and G. Harman, eds., Semantics of Natural Language, pp. 169-218. Dordrecht: Reidel.

Lozano-Pérez, T., and Wesley, M. 1979. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10), 560-570.