Massachusetts Institute of Technology
Artificial Intelligence Laboratory

A. I. Memo 1050a

Revised October, 1989
Original Version September, 1988

What are plans for?

Philip E. Agre
David Chapman

Abstract

What plans are like depends on how they're used. We contrast two views of plan use. On the plan-as-program view, plan use is the execution of an effective procedure. On the plan-as-communication view, plan use is like following natural language instructions. We have begun work on computational models of plans-as-communications, building on our previous work on improvised activity and on ideas from sociology.

To appear in Pattie Maes, ed., *New Architectures for Autonomous Agents: Task-level Decomposition and Emergent Functionality.* MIT Press, Cambridge, Massachusetts, 1990.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-85-K-0124.

Copyright © 1989 Philip E. Agre and David Chapman.

1 Introduction

What plans are like depends on how they're used. We contrast two views of plan use. On the plan-as-program view, plan use is the execution of an effective procedure. On the plan-as-communication view, plan use is like following natural language instructions. We have begun work on computational models of plans-as-communications, building on our previous work on improvised activity and on ideas from sociology.

The plan-as-program view and the plan-as-communication view offer very different accounts of the role of plans in activity. The plans-as-programs view gives plans a central role. Plan use is only a matter of *execution*, performed by a simple, fixed, domain-independent "interpreter." Plans-as-programs directly determine their user's actions.

The plan-as-communication view gives plans a much smaller role. It requires an account of *improvisation*. Plans, on this account, do not directly determine their user's activity. Indeed, an agent can engage in sensible, organized, goal-directed activity without using plans at all. An agent who does use a plan-as-communication does not mechanically execute it. Instead, the agent uses the plan as one resource among others in continually redeciding what to do. Using a plan requires figuring out how to make it relevant to the situation at hand, a process of interpretation which can be arbitrarily complicated and draw on a wide variety of resources. The plan-as-communication view portrays people and robots as participating in the world, not as controlling it.

Section 2 of this essay describes the plan-as-program view and some of the difficulties with it. The difficulties concern the computational complexity of plan construction, the problem of prediction in a world of uncertainty and change, the necessity of accommodating the simplicity of executives by specifying plans in impractical detail, and the largely unaddressed issue of relating plan texts to concrete situations in the world. This paper is not intended as a thorough survey of the literature on planning; for some useful surveys see (Chapman 1987), (Swartout 1988), and (Tate 1985).

Section 3 outlines our view that everyday activity is improvisatory in nature. Improvisation might involve ideas about the future, but in any event it requires a continual redecision about what to do *now*. Supporting this process of continual redecision is a technical problem that we have addressed in our work. We briefly describe Pengi, a system that employs novel kinds of perception and representation in playing a video game called Pengo.

Section 4 presents the plan-as-communication view and contrasts its views of plan use, representation, and activity with those of the plans-as-programs view. It further illustrates the view with an example of plan use in the real world. Our analysis of this example turns up several ways in which the plan's maker counted on the understandings of everyday reality that he shared with the plan's users.

Section 5 pursues this theme in more detail in relation to an ongoing project. Chap-

man is constructing a system, Sonja, which uses instructions given in the course of video-game playing. We argue that plan use and instruction use are similar, and sketch some of Sonja's capabilities.

Section 6 summarizes our principal conclusions and proposes that future inquiry conjoin computational analysis and model-building with principled and detailed observation of human plan use in natural settings.

Since we wrote the first version of this paper in 1987, several authors have proposed alternatives to classical planning. An appendix discusses a few of these proposals and compares them to our own.

2 Plans as programs

The plan-as-program view takes plan use to be like program execution. Almost all implemented executives have been modeled on programming language interpreters. A plan language, on this view, is like a programming language. The plans are built from a set of parameterized primitives (such as PUT–ON(x,y)) using a set of composition operators (to indicate serial execution, for example). Executing a plan means walking over it in a "syntactic," "mechanical" fashion, carrying out its primitive actions and monitoring conditions specified by the planner, and performing little or no new reasoning about the activity in which the agent is engaged. The executive is domain-independent: it applies no domain knowledge except that implicit in the plan and the machinery that implements its primitive action types. It makes no interpretations of its sensor inputs except for the monitored conditions and any predicates that might appear in plan conditionals. Nor does it second-guess the planner by performing any interpolations, substitutions, or rearrangements that would count as a departure from the plan. If the executive gets into trouble, it gives up and returns control to the planner.

(The plan-as-program view implies domain-independent *plan execution*, not domain-independent *plan construction*. Plan-as-program construction can be domain-independent or domain-dependent, algorithmic or case-based, formally correct or heuristic.)

This section discusses four reasons to doubt the plan-as-program view. (1) It poses computationally intractable problems. (2) It is inadequate for a world characterized by unpredictable events such as the actions of other agents. (3) It requires that plans be too detailed. (4) Finally, it fails to address the problem of relating the plan text to the concrete situation. These problems with the plan-as-program view do not mean that it is useless. We believe it may lead to practical applications in certain domains. Our arguments only apply to domains, such as the world of everyday life, that are relatively large, uncertain, and changing.

(1) The plan-as-program view makes planning into automatic programming with all

its formal undecidabilities. Chapman (1987) has proven several negative complexity results, both about the manipulations that need to be performed on partially specified plans and about the spaces through which plan-as-program planners must search. As formalizations of actions and preconditions become more realistic, these results get worse quickly. Complexity theory is, unfortunately, not an ideal tool for proving negative results. The appendix will briefly discuss the possibility that heuristic solutions might work well enough in practice. It is also possible that certain domains have enough useful structure to permit the construction of tractable domain-specific planners. Chapman (1987, page 353) has suggested that plan-construction algorithms at intermediate levels of specialization might capture varieties of formal structure that might be shared by classes of domains.

(2) The original planners made plans to achieve goals in well-behaved simulated worlds. In these imaginary worlds, it was possible to construct a plan which consisted of a representation of a sequence of primitive actions, which, performed in order, would provably achieve the goal. Thus it was possible to formulate the "planning problem" in terms of constructing something that would, when executed, *control* the robot. It has been widely recognized in the last few years that in the real world, execution of a plan brings important risks because unpredictable external processes can change the world and causally affect the robot.

Plans-as-programs are not very flexible. If the robot's interactions with its world fail to work out as the planner expected, the plan itself will fail. If the planner explicitly anticipates a specific, detectable uncertainty, it can either provide the plan with a conditional branch or it can fashion a strategy that works regardless of how the uncertain matter turns out. If the planner fails to anticipate an uncertainty, then a new plan will be required. Reasons to abort or revise a plan can be divided into two classes, contingencies and opportunities. If you're about to walk through the kitchen door to fetch a pen, a closed door is a contingency and a pen on a desk just outside the kitchen is an opportunity. Not all contingencies can be detected through precondition failures: maybe you can put your pants on over your shoes without violating any preconditions, but usually it is not sensible. Opportunities are still harder to test for because they are less obtrusive. An enormous range of circumstances might count as opportunities in one situation or another. In short, a new plan is called for whenever it is no longer sensible to continue following the existing one. This is a grave problem for any executive that does not share the knowledge and reasoning abilities of its planner.

In recent years, more attention has been given to the process of execution. As a result, executives have become increasingly complicated. Much of this complexity has resulted from practical experiences in trying to make plans-as-programs control real robots, experiences that suggest increasing the responsibilities of the executive. In the end, we believe, arbitrary amounts and varieties of domain knowledge can bear on how a plan

should be used. This means that an executive must have access to the full cognitive resources of the whole agent. If the full cognitive resources of the agent are devoted to using a plan, there will no longer be a separate executive module whose responsibility that is. The distinction between planner and executive can be eliminated; the whole agent is responsible for both plan making and plan use.

(3) It is generally acknowledged that no system could produce completely detailed plans in domains of realistic complexity. Real activity is too complicated for that. It follows that an executive has to be expected to fill in some details as it goes along. It also follows that a planner needs some idea of what details it can rely on its executive to fill in. A very simple executive will need everything spelled out for it, but if the executive had more knowledge and reasoning abilities then the planner could paint the desired actions with a broader brush. Ideally, a planner would only have to deal with issues that the executive itself cannot. Its plans would not be laden with redundant details. Nor would they prejudge decisions better left to the executive, which after all can base its judgments on the world as it actually turns out, not on models of projected worlds.

The plan-as-program view offers us one account of how a plan can be operational without spelling out every detail. If plans are like programs, then we can make compact plans using a hierarchy of subplans. The planner has a library of subplans, each of which has a contract. These contracts establish a partition between the issues that must concern the planner and the issues that subplans can deal with themselves. They enable the planner to live in a simple, abstract world, reasoning with the preconditions and effects of the top-level subplans.

This subplan-hierarchy view of plans has a number of shortcomings. First, the executive still cannot depart from the plan other than to return control to the planner. Second, it is unclear what sorts of domains permit hierarchical abstraction. The library subplans have to satisfy their contracts regardless of the specific circumstances; this makes them very difficult to construct. In complex real-world domains, where enormous numbers of concrete contingencies can bear on abstract goal ordering issues, truly hierarchical decomposition may not be possible. Lozano-Pérez and Brooks (1985) discuss the case of a robot manipulation task: the choice of the initial grasp, the motions of the hand required to position the part, the existence of a path to the destination, and the angles at which compliant forces will be applied are all tightly interconstraining. Moreover, the inevitable sensing and motion errors will propagate from one to another non-locally.

(4) An executive has to establish a causal connection between the text of the plan it is executing and the materials in the concrete situation in front of it. The ontologies of most existing plan languages posit a world made up of individuals, some of which correspond to constant symbols in the agent's axiom set. Thus, for example, the truth of a typical blocks-world proposition like ON(A,B) is determined by a relation corresponding

to ON applied to individuals corresponding to A and B. A plan might achieve the goal ON(A,B) by executing an action like PUT–ON(A,B). This requires that the executive be able to determine automatically which individuals in its world correspond to the constant symbols A and B. If every object has a bar-code affixed to it then it's easy enough. But blocks on tables and luggage in airports and cars in parking lots and turns on highways very often take work to distinguish. Arbitrary domain knowledge can, and regularly does, enter into determining which object is the one you want.

The practice of allowing primitive actions to traffic in constant symbols hides not only this problem, but a deeper one as well. Much of the work of using a plan is in determining its *relevance* to the concrete situations that occur during the activity it helps to organize. By hiding this work, an executive that can automatically relate symbols to objects, we feel, falsifies the nature of plan use. Using a plan requires domain-specific skills that a programming language interpreter does not possess and situation-specific improvisations that a programming language interpreter cannot perform.

The difficulties with the plan-as-program view reflect what we feel is a mistaken notion of what plans are for; that is, of what activity is like and of what role plans can play in it. The plan-as-program view understands activity as a matter of problem solving and control. The world presents an agent with a series of formally defined problems that require solutions. A planner produces solutions to these problems. The executive implements these solutions by trying to make the world conform to them.

The world of everyday life, however, is not a problem or series of problems. Acting in the world is an ongoing process conducted in an evolving web of opportunities to engage in various activities and contingencies that arise in the course of doing so. Most of what you do you already know how to do, and most of the rest you work out as you go along. The futility of trying to control the world is, we think, reflected in the growing complexity of plan executives. Perhaps it is better to view an agent as *participating* in the flow of events. An embodied agent must *lead a life*, not *solve problems.*

We suspect that part of the appeal of the plan-as-program view derives from the word "execution." To execute a command or instruction is to carry it into effect; to execute an action or operation is to perform it. The word is little used except in legal and administrative senses and by football coaches, but even its broader use suggests an activity that takes place in a narrowly specified institutional context, with articulated constraints and strict criteria, and with negligible room and need for variation, interpretation, improvisation, or any other deviation on the part of the person doing the executing. To "execute" a plan isn't just to "follow" it, it's to follow it "to the letter" and "by the book." In short, the word "execution" suggests thinking of a plan as a pretty-well-thorough representation of a sequence of actions, so that execution is a simple process. Perhaps that is why it has seemed so natural to assimilate plan execution to running a program on an programming language interpreter.

3 Participation

What might an alternative to plans-as-programs look like? Let us start by retiring the prejudicial term "execution" and instead speak more neutrally of people (or robots) as "using" plans. This simple terminological change makes some hard questions seem more urgent. First, what can one do with a plan besides executing it? Second, how do plans and plan-making change if plan users can be counted on to use plans sensibly rather than mechanically executing them?

We don't know if these questions must have the same answers for robots as they do for people. But so long as alternatives to the plan-as-program view are in short supply, evidence from human plan use can bring some perspective. Most of what is known has been discovered by social scientists such as Gladwin (1970), Hutchins (1987), Scher (1984), Suchman (1986, 1987), Scribner's group (Scribner 1984, Beach 1986), and the Soviet activity theorists (Vygotsky 1962, Wertsch 1985). Our own analysis draws on this material, though a detailed discussion of it would take us too far afield here.

The plan-as-program view gives plans a central role in determining activity. In particular, it claims that an agent acts as it does because it has a certain plan. We do not believe this claim. According to the plan-as-communication view, a plan does not directly determine an agent's actions. Instead, a plan is a resource that an agent can use in deciding what to do. What, then, *does* determine an agent's actions? Answering this question is the job of a *theory of activity.* After briefly summarizing our understanding of activity in this section, we will return to the question of the role of plans in activity.

Our theory of activity has two interconstraining parts: a theory of cognitive machinery and a theory of the *dynamics* or regularly occurring patterns of activity. In studying people we ask (i) how is ordinary human activity organized and (ii) what does this imply for the organization of human cognitive machinery? In studying machines we ask (i) what forms might an agent's activity take and (ii) what sorts of cognitive machinery are compatible with what sorts of activity?

Our answers to these questions are informed by the central theme of participation in ongoing activity whose determination is shared with other processes and agents. Everyday routine activity, we believe, has an orderliness and coherence that is independent of any plan or other representation of it. See (Chapman and Agre 1986) for some of this story and (Agre 1988) for some of the rest. We have found, in the case studies that we have conducted, that participating in the flow of the environment, rather than attempting to control it, can simplify the machinery required to account for the organization of activity.

We built the Pengi system (Agre and Chapman 1987) to illustrate some of what we have learned. (For details about Pengi's workings see Agre 1988.) Though Pengi engages in complex patterns of activity, its machinery is extremely simple: a visual system based on psychophysically motivated ideas from Ullman's visual routines theory

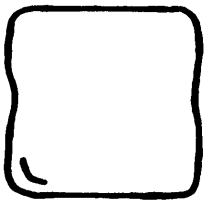
(1982), a simple motor system, and a central system made entirely of combinational logic. (For some related projects, see Brooks 1986, Drummond 1989, Kaelbling 1986, Maes 1989, Nilsson 1989, Rosenschein and Kaelbling 1986, and Schoppers 1987.)

Pengi does not follow any plans, but neither is it pushed around by its world. The Pengo games Pengi plays move fast, so Pengi constantly uses the contingencies and opportunities of its environment to help it improvise ways to pursue its projects. Improvisation differs from planning-as-programming in that each moment's action results, effectively, from a fresh reasoning-through of that moment's situation. Yet improvisation, like planning, involves ideas about what might happen in the future.

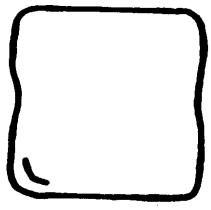
One of Pengi's contributions is a new participatory theory of representation we call *indexical-functional*, or *deictic* representation (Agre 1988). Whereas traditional representations posit a "semantic" correspondence between symbols in an agent's head and objectively individuated objects in the world, our theory describes a causal relationship between the agent and *indexically* and *functionally* individuated *entities* in the world. For example, one of the entities Pengi works with is *the-bee-I-am-chasing*. This entity is individuated indexically in that it is defined in terms of its relationship to the agent ("I"). It is also individuated functionally in that it is defined in terms of one of the agent's ongoing projects (chasing a bee). Whereas in a traditional representation, the symbols BEE-34 and BEE-35 would always refer to the same two bees, different bees might be *the-bee-I-am-chasing* at different times. Pengi uses its visual routines—patterns of directed visual activity—to register *aspects* of various entities, for example *the-bee-I-am-chasing-is-running-away*. The participatory nature of deictic representation means that Pengi deals with its environment through a constant interaction with it rather than through the construction and manipulation of models.

Let us consider a relatively complicated example, starting from the situation illustrated schematically in Figure 1. In this situation, the penguin (played by Pengi) wants to kill the enemy bee by kicking an ice cube at it. Ice cubes, when kicked, slide across the two-dimensional game board in a vertical or horizontal direction. Thus, to kill a bee, an ice cube must be aligned with it in one of the two Cartesian dimensions. In this situation, no ice cube is aligned with the bee. However, if the penguin first goes over to the ice cube labeled *the-projectile-cube* and kicks it right, it will collide with the ice cube labeled *the-stop-cube* and come to a halt. (Energy is not conserved in this game.) *The-projectile-cube* will then be aligned with the bee, and can be kicked at it.

A planning system might approach this situation by constructing a four-step plan: go to left side of *the-projectile-cube*; kick *the-projectile-cube*; go to top of *the-projectile-cube*; kick *the-projectile-cube*. The executive that is given this plan must verify the plan's continued applicability by checking a long list of conditions that might have arisen to invalidate it: the bee might wander away, or another bee might draw close and need to be dealt with, or another bee might kick some ice cubes and thereby disturb the configuration in a way that makes carrying out the plan impossible.



the-projectile-cube



the-stop-cube

the-penguin

the-enemy-bee

Figure 1. A Pengo situation that requires looking ahead.

Pengi constructs no plans and no models of hypothetical future worlds. It does, however, take probable future circumstances into account; in place of simulation Pengi uses *visualization.* Pengi *looks to see* what might happen next. It engages in visual routines which find particular spatial configurations that predict courses of events and so suggest actions. For example, when Pengi sees that an ice cube adjacent to the penguin is aligned with a bee, and there are no intervening ice cubes, it kicks it, making it likely to strike and kill the bee. When it sees such an ice cube that is only near, rather than adjacent to, the penguin, it moves the penguin in the direction of the ice cube, because once it gets there the bee might still be aligned. If no ice cubes are aligned but the complex configuration of Figure 1 obtains, Pengi sends the penguin over to *the-projectile-cube* in order to kick it at *the-stop-cube.*

Put in the situation of Figure 1, Pengi may well engage in the same course of activity a planning system would, but for quite different reasons. Consider, for example, why each system would take the fourth and final action, kicking *the-projectile-cube* at the bee. The executive would take this action because the value of its program counter is four. Pengi takes the action because, by visualizing, it can see that by doing so it is likely to kill the bee. Once it has gotten to that point, it has no use for the idea that kicking that ice cube is part of a larger pattern of activity.

Even though Pengi's network is only partially implemented, it still plays a pretty decent game of Pengo. We started designing the network by envisioning a series of scenarios, which we call *routines*, of the common patterns of interaction between the player and the game. In practice, Pengi often exhibits these routines. It is relatively unusual, though, for Pengi to carry off one of our envisioned routines without a hitch. Pengi regularly aborts a routine when a contingency arises, tries some action repeatedly until it works or until it notices a more promising option, embarks on a new routine when an opportunity arises, interleaves different routines, and combines its repertoire of activities in useful ways we didn't anticipate. (It also regularly does silly things in situations for which we haven't yet wired it.) Pengi exhibits this flexibility precisely because we did not convert our envisioned scenarios into a repertoire of plans for Pengi to execute. Instead, we analyzed what reasoning should lead Pengi to take what actions in what kinds of situations. When it was possible for these lines of reasoning to conflict, we implemented a simple arbitration scheme that takes various aspects of context into account in deciding which of the various plausible actions to take on each next moment. Pengi thus effectively makes fresh decisions on every cycle of its clock about the nature of its current predicament, what goals it should adopt to deal with that predicament, and what actions it should take to further those goals.

The plan-as-program view may make it a little difficult to grasp the sense in which Pengi creatively improvises. It may seem that, since Pengi will always do just what it's wired up to do, it is completely *un*creative; no more than a wind-up toy. This is, however, equally true of a planner: it too does just what it was programmed to. The

apparent difference lies in the fact that a planner explicitly considers and rejects many alternative actions. This seems to give it an infinitely generative combinatorial power that Pengi lacks. Pengi, however, has its own kind of generativity, an infinity of dynamic possibilities rather than an infinity of structural combinations. Like Simon's Ant (Simon 1970), Pengi enters into forms of interaction with its environment which exhibit a great deal of complexity, but for reasons which do not lie in either Pengi or its world alone. Like a planner, Pengi is "given" something and complexity arises from the combinatorial possibilities that arise when that something is put into action. With a planner, these combinatorial possibilities are explored mentally. With Pengi, they are encountered in the course of the activity. We did not design a device that could *simulate* a certain complex way of life; we designed a device that could *live* that life.

It is sometimes said that we should measure a system's achievement by comparing how much it is "given" as compared to how much it "does for itself." By this criterion, though, it is not obvious whether we should prefer Pengi or a conventional planner. Pengi has an invariant part of its architecture (the visual system and the general central system design) and a part varies with the domain (the wiring of the central system). A planner, similarly, has an invariant structure (the plan-space search algorithm) and a part that varies with the domain (at a minimum this includes a set of operators, an axiomatization of the domain, and a set of primitive sensors; to be efficient, it will probably also require a set of domain-specific inference mechanisms and search strategies). It is hard to say which kind of system has the better ratio of "given" to "done-for-itself." Since a real autonomous agent should be able to learn new domains for itself, a real comparison will depend on which is easier to learn; and in the absence of well-worked-out theories of learning for either alternative this is as yet impossible.

The circuitry in Pengi's central system, of course, has a fixed structure. Designing this circuitry is as hard as any other sort of complex programming; we do not believe that such circuitry could arise through general-purpose automatic synthesis. Instead, we believe that this circuitry arises in the course of the agent's interactions with its environment, through a long process of incremental evolution. In earlier work on *running arguments*, Agre (1985, 1988) investigated some of the forms this process might take. He built a fairly general-purpose rule system, a streamlined version of Amord (de Kleer *et al.* 1977), which maintains dependencies on its reasoning. As it runs, it accumulates a large dependency network that can be regarded as a combinational circuit connecting perception and memory elements to primitive actions. Though not entirely realistic, this scheme provides some sense of the form that incremental learning through modification of central-system circuitry might take.

The running argument system's dependency network grows in the course of its interactions with the concrete situations in which it must decide what to do. Faced with any situation at all, the system behaves as though it has forward-chained all of its rules to

exhaustion. It does not achieve this behavior, though, by continually undertaking the very expensive process of actually running all those rules. Instead, most of the work is actually done by the previously conducted lines of reasoning stored in the dependency network. The system only runs the new rules that the dependency network does not record, so the amount of actual rule-firing effort that a new situation requires is proportional to how novel it is. Once the running argument system has encountered the normal run of situations that occur in its environment, it needs to run very few rules at all. Since "situations" are counted in terms of their significance for the agent's current understandings and goals and not in terms of an exhaustive enumeration of true propositions, the system's recorded lines of reasoning will apply to a wide variety of future circumstances.

In forms of activity which are generally routine, such a system will be able to engage in rapid, flexible forms of interaction without a prohibitive computational overhead. The principal shortcoming of the particular system described in (Agre 1985, 1988) is that it is incompatible with a representation scheme based on constant symbols; this was one of the motivations behind the development of deictic representation. The general strategy of learning through particular occasions of situated reasoning about action is similar in spirit with the notion of case-based planning (Hammond 1989), whose conception of memory structure is much more elaborated.

Pengi illustrates some ideas, but Pengo-playing differs from other human activities in many ways. Most activities are less hectic, have more complex goal structures, require more remembering, involve additional kinds of representation such as visual imagery and internal language, and so forth. Our experience with Pengi has focused the issues for a new round of study of dynamics and machinery.

Pengi, as we have mentioned, neither makes nor uses plans. Pengi engages in a continual, participatory interaction with its environment. Yet its activity is directed toward particular concrete goals: killing certain bees, staying clear of others, becoming adjacent to ice cubes it might usefully kick, and ultimately winning the game. Does this mean that plans are useless? Not at all. Pengi is a study of a certain subset of the dynamics of improvisatory activity. A creature that can participate in this set of dynamics can play Pengo.

Many other activities do require plans. If Pengo got harder, for example, Pengi might sometimes have to refer to a plan. The plan might explain how to deal with some tricky situation, or perhaps what strategic issues bear on the matter of which bees to attack when. But what would be involved in using plans like these? What role can plans play in an improvisatory theory of activity? This is the subject of the remainder of the paper.

4 Plans as communication

In place of the plan-as-program view, we would like to propose a different account of what plans are for which we will call the plan-as-communication view. The two views differ as to the nature of plan use, the way in which plans are representations, and the nature of activity.

Nature of plan use. For the plan-as-program view, a plan decomposes into primitive actions which can be simply "emitted" by the executive, a simple, fixed, domain-independent device. For the plan-as-communication view, figuring out what activity a plan suggests requires a continual interpretive effort. It can take a lot of work to determine what in the situation the plan is talking about. A plan is operational if a sensible and suitably acculturated agent can use it, somehow, to engage in the activity it describes. A plan is a resource you can draw on in deciding what to do, on an equal basis with other resources such as the arrangement of your equipment, external memory devices like scratch paper, and help from your friends (Suchman 1987). Unlike executives, people using plans know more or less what they are doing and why. Thus a plan is often well thought of as a mnemonic device. (For another computational interpretation of the idea of plans as resources that, unlike our own notion, maintains a notion of a complete world model, see Payton 1989.)

Nature of representation. A plan-as-program "represents" a course of action in a very simple sense, insofar as programming languages have roughly compositional semantics. Each primitive of a programming language always occasions the same action, regardless of the context. For the plan-as-communication view, a plan "represents" a course of action in a much more complex sense, insofar as a linguistic entity's meaning depends on the context of its use in a hundred different ways. In particular, a program represents its actions "exhaustively" where a linguistic entity cannot and need not.

On the plan-as-program view, plans are abstract mathematical entities. On the plan-as-communication view, plans are social constructions (Hutchins 1987, Wertsch 1985). Children learn collaboration before they make plans for themselves. Our ability to make and use plans is built on our ability to use language during activities we share with others. Many plans are physical objects such as wall charts, instruction sheets, blueprints, and bound business plans. Others are spoken utterances, as might be produced in response to the question "So, what's your plan for the afternoon?" The nature and use of these externally represented plans are relatively easy to study. People also make and use plans that are represented only internally. We'll suggest later that these are similar to external ones in important respects. In any case, external plans seem like a good place to start study.

Plans-as-communications, unlike plans-as-programs, do not constitute a unified phenomenon. A lot of disparate sorts of things, used quite differently, can count as plans-as-communications. Plans-as-communications shade off into a variety of other phenomena,

such as mnemonics and conversation and visual imagery and written lists and schedules and workspace arrangement. While you might be able to implement an agent that could use plans-as-communications, the agent's design would not contain a plan-as-communication-using module, since the whole agent must be brought to the task.

Nature of activity. In the plan-as-program view, the only situation given thorough consideration is the "initial situation" passed in to the planner. During the course of execution, the circumstances that arise can only determine conditional branches or cause control to be returned to the planner if something goes obviously wrong.

The plan-as-communication view is part of a theory of "situated activity" (Suchman 1987, Lave 1988). Situated activity is not some special variety of activity. The phrase emphasizes that a central feature of *all* activity is that it takes place in some specific, ongoing situation. The plan-as-communication view suggests that the world's independence of your control is not an obstacle to be overcome but a resource to be made use of (*cf.* Suchman 1986). If your activity is not rigidly controlled by a plan, contingencies need not be disruptive; instead they can occasion creative improvisation.

In choosing the plan-as-communication view over the plan-as-program view, we implicitly promise to explain the role of plans-as-communications in a broader theory of situated activity. This is a big project. The remainder of this essay sketches some starting points.

Let us consider a typical example of human plan use. The route from my (Agre's) flat in Boston to the subway station, a distance of about three blocks, is hard to describe without drawing maps. (See Figure 2.) Nonetheless, we found that three experimental subjects unfamiliar with the area had no difficulty traversing the route using as a plan only the written instructions "left out the door, down to the end of the street, cross straight over Essex then left up the hill, take the first right and it'll be on your left," which is nothing compared to the actual complexity of the trip.

Consider how much these directions leave out. "The door" is presumably the front door of the building. There's no need to tell you to walk down the street in the direction that "left out the door" will leave you headed; when you're on a path you don't need a plan. No need to label "down to the end" a figure of speech rather than an instruction to descend somewhere. No mention, either, of the fact that Essex Street is not marked as such anywhere near its intersection with Edinboro Street. There's no need to mention it, since it'll be clear which street is meant once you get there. (Our subjects reported being bothered by the lack of a sign but all of them proceeded correctly anyway.) "Left up the hill" will manage to refer to the Avenue de Lafayette rather than to Essex Street because it's the only hill you can see when you're standing at that intersection looking that way. Getting to the Avenue will require a brief rightward detour to get around a fence. No need to mention either this detour or the necessity of crossing the Avenue.

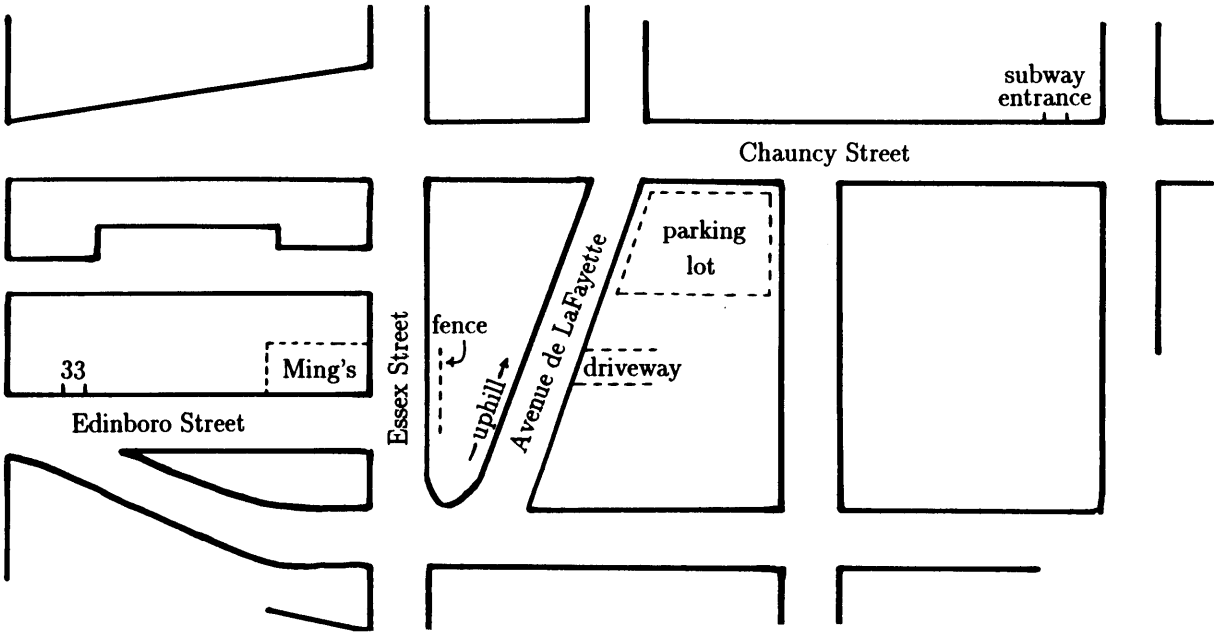


Figure 2. The route from 33 Edinboro Street to the Washington Street subway station, early 1986. (Not to scale.)

When I walk this route myself I typically cut through a parking lot that precedes the "first right." The directions leave out the parking lot altogether; presumably you will have the sense to see the first right coming and cut the corner; and it doesn't matter if you don't. You'll also need the sense not to interpret a driveway or the parking lot itself as that first right. Everyone relies heavily on these sorts of things, usually without specifically knowing it, when giving directions. Some people are better at it than others. For example, experienced urban direction-givers know that alleys can confuse people who've been directed to count lefts or rights.

When you're using a plan, your surroundings are available as a resource for interpreting it. A plan that refers to "the hill" counts (roughly speaking) on there only being one hill apparent to someone who has gotten that far in the plan. A plan that instructs you to "take the first right" counts on it being clear which street is indicated. "Counting" and "clarity" are defined reflexively, almost circularly, as that which a given person will be able to figure out in a given situation.

The plan also relies on your experience and skill. The instruction to "walk down to the end of the street" assumes you have the sense to disobey it when the street is full of slush or garbage or dangerous-looking people, as it often is. The plan omits things you already know, like how to cross a street, how to use street signs, how to detect another street coming up, and where it's safe and legal to walk. It also omits things you can

be trusted to figure out for yourself, like how to recognize the subway station, how to wind your way past the trash strewn outside Ming's grocery, and how to get some new directions if you get lost.

In short, this plan exploits a long list of ways in which its maker and its user share an understanding of the world. We would like to suggest that this lesson generalizes in several ways: that the list of shared understandings is actually innumerably long; that all plans depend on shared understandings in this way; that action in the real world is sufficiently difficult to specify that plans *must* depend on innumerable shared understandings to be expressible at all; and that all of these points apply regardless of whether the plan's maker and user are the same agent or different agents.

The plan-as-program complexity analysis does not apply to plans-as-communications because they are guides to activity, not solutions to problems. This does not, of course, mean that making plans-as-communications will be easy. One important factor that ought to simplify the process, though, is the knowledge that the plan's user will use it intelligently rather than executing it like a computer program.

5 Sonja

Reducing plan use to natural language comprehension might not sound very helpful. We certainly don't want to trivialize the role that natural language plays in situated activity; it's a big topic (see for example Heritage 1984, Stucky 1987). We have simplified the problem of plan use by studying *situated instruction use*, the use of instructions given in the course of on-going activity. Situated instruction use is analogous to plan use, with three principal differences: that instructions are given at appropriate times, whereas a plan user must keep track of where it is in the plan; that situated instructions are typically simpler syntactically than are full-fledged plans; and that instructions are provided by an external agent, whereas plans may be made by the same agent that uses them. The first two of these differences are straightforward simplifications; the third requires some comment.

External plans you've made for yourself, such as written-down lists of things to do, are perhaps more obviously similar to those made by someone else. The principal difference is that in making a plan for yourself, you can leave out still more detail and be idiosyncratic about abbreviations and conventions. The plan you make for yourself will function more as a mnemonic device than, say, a recipe in a new cookbook. Nonetheless, the same issues of keeping track of the plan's relationship to the concrete circumstances will come up.

We take instructions in internal language (that is, silent speech conducted in ordinary natural languages) as prototypical of internal plans-as-communications. We take literally the phrase "telling yourself what to do." We believe that the ability to instruct

yourself in this way is based on your ability to give instructions to others and to follow instructions given to you. This stance has been explored by the psychologist Vygotsky (1962), who explains the similarities between internal and external language and also how internal language becomes abbreviated and idiosyncratic. In (Chapman and Agre 1986) we suggested that this internalization process works by part of your brain coming to simulate the external world, thereby "fooling" other parts of your brain into thinking that they are actually engaging in activity when you are actually just thinking; a similar account appears in (Rumelhart *et al.* 1986). In this case, the necessary hardware simulates the process of hearing what you are saying out loud, forming a sort of "null modem" that lets you hear what you are thinking.

Situated instruction use differs from logical advice-taking (McCarthy 1958) in that the agent interprets the instructions in relation to its specific ongoing situation. Instructions are understood in terms of the agent's existing ability to understand what is at stake in the situation and to act autonomously to pursue its projects. The instructions thus play only a *management* role.

Chapman (forthcoming) is now constructing a system called Sonja[1] which, like Pengi, can engage in complex activity without use of plans, but which can also use instructions when given them. Sonja is based in part upon an empirical study of video tapes of human video game players who are given advice by a kibitzer, or by another player when two are playing cooperatively. The players in these tapes are already good at video games and at the coordination required for cooperative play; in many cases they are expert at the particular game they are playing. As a result, their activity is largely routine. Moreover, the players see the same screen and have much the same understanding of the game, so they can depend on their shared understanding to achieve most coordination. Thus they need say very little. With rare exceptions, their talk serves only to repair minute differences in understanding. One player might simply say "No!" because there are only two activities the other might plausibly undertake in the current situation. The utterance exploits their commonality of understanding to interpret the listener's moves as constituting a certain activity, judge that activity to be the wrong one, and suggest that he desist from that activity and instead join the speaker in the other one.

To take another example, very often on our tapes one player will say to the other "Turn left!" Most often, the other player does not immediately turn left. Yet this is not an error, nor is the advice erroneous, nor does the speaker consider that she has been disobeyed. In fact, a viewer will generally agree that the instruction was carried out. Activity other than immediately turning left can count as fulfilling the instruction in many domain-specific ways.

- In some cases, the doorway through which it will be possible to turn has not yet

[1]Sonja is pronounced with an English *j*, "Sahn-djuh," not a Continental one, "Sewn-ya," because it is named after a comicbook character.

been reached, so that turning left would run you into a wall. In these cases, turning left is deferred.

- When the point at which a turn is possible is reached, there may also be a doorway on the right, and there may be an monster hiding behind the door. If the monster will shoot her in the back when she turns left, the player will turn right and kill the monster before turning back around and proceeding.
- In one case in our collection, the player passes the turn to pick up a valuable energy pod and then returns to comply with the instruction.
- Again, it may be that there is no left turn available, but there is an obviously correct right turn; in this case, the player may well figure that her interlocutor has simply said "left" for "right" in the heat of the moment, and turn right without comment.

The player is only likely to say "huh?" when completely unable to make sense of the instruction.

Not only can instructions be deferred; often they can be enacted with actions that, taken literally, violate them. For example, during a game of Gauntlet one player said "Don't go below that line," pointing at an imaginary line on the screen. Monsters in Gauntlet always head straight for you. Thus it is often important not to pass below the edge of a wall; if you do, monsters will stream around the corner and attack you. However, everyone eventually *did* go below that line without the instruction being explicitly rescinded; they mutually understood that it was now time to go after that particular set of monsters.

Videogame instruction can be so compact because their possible import is heavily constrained by *indexicality, projection,* and *reflexivity.*

Indexicality. We interpret communications with regard to the present circumstances. "No" offers advice about some ongoing activity whose manifestations are visible to both players through the motions of one of the figures on the screen. "Turn left" picks out a certain corridor in the maze, one which is specified in terms of the listener's current location and heading. "Don't go below that line" picks out a certain *imaginary* line that the speaker can point at because both parties know to visualize it. In each case, the players are not making reference to objectively available "features" of the video screen but to shared interpretations of the commonly-visible whirl of colored lights.

Understanding indexical instructions involves complex perceptual processing, which is carried out using Sonja's visual system. (This visual system is similar to, but more sophisticated than, that of Pengi.) This processing often results in a new take on the situation. For example, if Sonja hears "Use the knife!" and it hasn't yet noticed any knives in the scene, it uses a visual routine to find one. Once it can see the knife, it notices properties of it, for example that it is easily accessible, and acts on them.

Projection. Each of us knows what might be expected to happen next. An imperative like "No," "Turn left," or "Don't go below that line" will typically invoke a projection of the specified course of events and another projection of the "or else" that might result if the listener disobeys. Skilled players will generally be able to perform both projections since they are familiar with the ways of the game. Visualization is Sonja's principal means of projection.

Reflexivity. Both the player and her advisor understand that they share an understanding of the situation; since the other person's understanding is part of the situation, this applies recursively. The kibitzer can only expect "No" to communicate if the player understands herself to be engaged in the particular activity "No" recommends against; the player can only make sense of the instruction if she imagines that the kibitzer considers her to be engaged in that activity; the kibitzer must further be able to count on the player imagining this; and so on. Likewise, both players must reflexively understand that "Turn left" picks out a certain corridor and that "Don't go below that line" picks out a certain imaginary line.

In our empirical studies, the players assume to an amazing extent that they both see the evolving game the same way, despite its large number of continually shifting issues. The players *must* make this assumption. If they didn't then they could never finish specifying everything that would be necessary to relate their advice to the evolving game situation. Indeed, we doubt if the players could list their shared understandings if they had to. Communication doesn't pick up a "meaning" from my head and set it down in yours. Instead, communication is part of the work of maintaining a common reality. The players share a common reality because they are competent players and because they use language to keep their shared reality in good repair.

Sonja illustrates certain themes in natural language pragmatics. It makes no attempt to implement a realistic theory of syntax or semantics. However, (Chapman forthcoming) will sketch a theory of syntax which does not require the creation of parse trees or other datastructures, and which may be implementable in a Pengi-like architecture. This theory, based on *linguistic routines*, might support plan making and might be amenable to internalization. It might, therefore, be the basis of an account of plan-as-communication making and use; but this is all still highly speculative.

6 Conclusion

We have outlined and contrasted two views of the nature of plans and plan use, the plan-as-program view and the plan-as-communication view. We have offered some reasons to doubt the plan-as-program view and speculated briefly about the nature of plans viewed as communications about situated activity. Specifically, we argued that

1. Plan use is not necessary for sensible action. When a plan *is* used, it does not directly determine its user's activity.

2. Plans represent the activity they describe in the way a recipe represents the activity it describes, rather than in the way a program represents the computation it describes.

3. Figuring out how a plan relates to your current situation requires a continual interpretive effort. This interpretation is often difficult and can require arbitrary domain knowledge and reasoning abilities. It can also require concrete actions such as looking around, asking for help, and manipulating the materials at hand to see how they relate to the ones mentioned by the plan.

4. The ability to make and use plans arises from, and is continuous with, one's experience with cooperative language use in the context of ongoing concrete activity.

5. Plan use relies on an unbounded set of assumptions that the plan's maker and user share concerning activity in the world generally and the evolving concrete situation in particular.

6. Using one's own plans is much like using plans communicated by someone else.

Many of the technical questions raised by the plan-as-communication view are as yet ill-defined, and certainly unanswered. Our initial ideas are only starting points. We do suggest, however, that research into the dynamics of plan making and plan use requires a worked-out view of the nature of everyday activity. Finally, we suggest that a critical and never-ending prerequisite to such an understanding is continual, detailed, sociologically informed observation of the ordinary everyday situated activity of the only truly successful plan makers we know of, namely human beings.

A Some other alternatives

Since we wrote the first version of this paper in 1987, several other papers have appeared reporting projects which address the difficulties that have come up with traditional planning ideas. We cannot conduct a comprehensive review here. Instead we will briefly discuss some of the work that goes by the names of *interleaved planning*, *behavioral modules*, *heuristic planning*, *reactive planning*, and *plan-as-constraints*. These proposals avoid many of the problems we have ascribed to the plan-as-program view, but some of the other problems remain for further research.

Recent interest in mobile robotics has focused attention on embodied activity in various ways. The first and most common response to the difficulties planner-based architectures face in actually acting has been *interleaved* or *incremental* planning (Chien and Weissman 1975, Giralt *et al.* 1984, McDermott 1978, Tate 1984, Wilensky 1983, Wilkins 1985 and 1988). In interleaved planning the planner makes its plan as always. When the executive gets into trouble, it passes control back to the planner, which assesses the situation and makes a new plan.

In order to implement interleaving, an executive needs some notion of when it is in trouble. In a perfect world, the executive would always be able to determine whether continuing with the current plan is the best thing to be doing. In reality, though, an executive has little or no access to the reasoning behind a plan, much less to the courses of action that the planner decided *not* to undertake. Instead, the executive will typically monitor a set of conditions that need to hold at various points during the plan, particularly the preconditions of the various plan steps. Such a system encounters two difficulties. The first is that it does not detect trouble until it has become relatively obvious. It is as if, when driving a car, one did not change direction until one has hit something. The second is that it does not provide a way of registering unexpected opportunities. The triangle table approach of Strips does allow the system to take advantage of opportunities that render planned actions redundant, but it does not help in detecting conditions that would have led the planner to construct a very different plan. This is why we argue that it is best to have as much as possible of the agent's reasoning power on-line, a conclusion that leads us to propose erasing the distinction between plan construction and plan execution and ultimately to a different notion of plans.

Interleaved planning and improvisation differ in their understanding of trouble. In the world of interleaved planning, one assumes that the normal state of affairs is for things to go according to plan. Trouble is, so to speak, a marginal phenomenon. In the world of improvisation, one assumes that things are not likely to go according to plan. Quite the contrary, one expects to have to continually redecide what to do. This is not to say that the resulting activity is chaotic in nature. It is, however, to say that the orderly nature of the activity, on whatever scale, does not arise from its having been mapped out ahead of time through the construction of a plan. Instead, the orderly nature of the activity

arises through the interaction of an improvising agent and that agent's familiar world.

We think of this distinction between interleaved planning and improvisation in terms of the theme of control. A system that operates by constructing and executing plans lives, to speak metaphorically, in a sort of fantasy world, the one it projects when it reasons about its future actions. In this way, the system believes that it has a kind of control over its world that, at least in many domains, is not realistic. When things do not work out as projected, the system is surprised. An improvising agent, by contrast, does not live its life through an alternation between fantasy and surprise. It does not believe that it has complete control over its world. Instead, through a continual give-and-take with its environment, creatively making use of opportunities and contingencies, it participates in the forms of activity that its world affords. On account of this contrast, we feel that the word "reactive" would be much better applied to interleaved planning systems than to improvising systems.

Proposed interleaved planners have all used standard planning techniques. As a result, interleaving the construction and execution of plans does not help with many of the shortcomings of the plan-as-program view. If plan construction is a computationally intractable process at the beginning of a task, it is also going to be an intractable process when trouble arises and a new plan is required.

Some current projects are trying to make plans-as-programs more accommodating of environmental variation using primitive actions, or in Malcolm and Smithers' terms "behavioral modules," that interact in complex but well-understood ways with the physics of the domain to effect specified conditions (Malcolm and Smithers 1989, Erdmann and Mason 1988). In both cases the actions slide objects of uncertain locations into specified positions without extensive use of sensors, but the point ought to generalize (see Miller 1989). This idea is closely related to the schemes through which the ethologically inspired robots built by Brooks and his group gain simplicity by relying on regularities in the physics of its interactions with their environment (Brooks 1989, Connell 1987). In each case, as in Pengi and Sonja, simplicity of machinery results from close attention to the dynamics of recurrent forms of activity.

Another approach to alleviating the shortcomings of classical planning is heuristic planning. This term actually could mean either of two things: it could refer to heuristics that would help search the space of plans, or it could refer a planner that is only heuristically correct (*i.e.*, sometimes produces incorrect plans).

In order to produce correct plans in a reasonable amount of time, a system must have some way of controlling its search through plan-space. The necessary techniques might be domain-independent or they might be domain-specific. We do not know of any powerful domain-independent heuristics for searching plan spaces. We suspect that none exist, in part because of the results which show domain-independent plan-construction

to be as computationally complex as computation in general.

We have more hope for research into plan construction in particular domains or classes of domains. No doubt many domains have structure which the plan construction process can exploit. This might take the form of heuristics that can recognize which plans are most promising before the planner has elaborated them very far (Hayes 1987), or of a plan representation which makes the search space inherently small, perhaps because of locality considerations (Lansky 1987). Such techniques might work well in certain factory automation tasks, for example, since they take place in a highly structured and constrained environment. Our own concern is with the design of agents that can carry on autonomously in worlds that have the very different kind of structure and constraint that characterizes everyday life. In these worlds, the case for plan-construction search heuristics remains to be made.

Another attractive possibility is a planner that does not always produce correct plans. This approach might circumvent the negative complexity results by posing the planner a less difficult problem. It also addresses the observation that getting along in the world does not require that you do the "right" thing all the time, just that you do well enough. Research on such systems must find a criterion of adequacy other than simple formal correctness. A natural criterion would be probabilistic correctness: the system might be good enough if provably most of its plans were correct. If this fraction is very high, the incorrect plans can be neglected. On the other hand, this seems like a very difficult criterion. People often set out to do something that turns out later not to make sense; at some point they figure out that they are doing the wrong thing and recover. This suggests an alternative criterion of adequacy, that the executive be reliably able to detect incorrect plans in time and to recover, on its own or by getting the planner to produce a new, provably correct plan. This would demand a lot of the executive. As with interleaved planning, a better approach would be to avoid the modularity of planning and execution, so that as much as possible of the system's reasoning power is available as a plan is used. That way, the agent is much less likely to keep executing a plan that's no longer sensible.

Another recent approach to activity is *reactive planning* (Firby 1987, Fox and Smith 1984, Georgeff and Lansky 1986 and 1987). The systems that have gone by this name have consisted of a conventional executive together with an externally generated plan library. Their aim is to act flexibly by having a repertoire of plans available and choosing among them as circumstances evolve. What these systems amount to in practice depends on how big the individual plans are. If the plans are very small, then it is hardly worth calling them plans; the selection mechanism itself is doing all the work that needs to be explained. If the plans are not very small, and especially if they are relatively large, it seems that all of the difficulties with the execution of plans-as-programs will apply to them as well. This ambivalence is already present in the phrase "reactive planning," which seems like a contradiction in terms. A theory of action must explain how action

can both have long-term purposes and take account of short-term conditions. Reactive planning systems seem not to resolve this tension: they can only take account of short-term conditions by pursuing the equally short-term goals of their individual plans.

In general, a great deal of dispute has surrounded the term "reactive." Pengi, for example, is often said to be a reactive system. We, however, never use the term ourselves because we feel that it enters into a mistaken opposition. The verb "to plan" takes on two different meanings in AI. The first, more general meaning relates to reasoning about action, and especially about how actions lead to goals, regardless of what form this reasoning takes or how it connects to whatever actions the agent eventually performs. The second, more specific meaning relates to the process of constructing plans in order to execute them. These meanings are distinct because constructing and executing plans might not be the only way of organizing purposive activity. Pengi, for example, reasons about actions but does not construct or execute plans. The word "reactive" usually seems to be opposed to planning in the *first* sense; under this usage, Pengi is not reactive. On the other hand, if "reactive" means neither constructing nor executing plans (the *second* sense), Pengi is in fact reactive. Notice, though, that the second, weaker sense of "reactive" is compatible with all of the phenomena that "planning" is supposed to explain, in particular the pursuit of goals, anticipation of the future, and the complex organization of activity in general. The question is one of what kinds of explanations of these phenomena are possible.

This paper seeks to introduce new terminology to clarify the distinctions that are being created as new ideas are introduced into research on activity. The noun "plan" was once quite well defined in AI; it meant a program, especially one that had been automatically synthesized, that was intended to be executed by a robot. We would like to retain this crisp and useful meaning for discussion, so we refer to it as plan-as-program. We have now proposed another meaning of "plan;" to avoid confusion, we refer to it as plan-as-communication.

We can understand the notion of reactivity in terms of loci of control. The plan-as-program view identifies a locus of control for activity in the planner. A chain of command descends from the planner to the executive to the world. We might speculate that "reactivity" means displacing the locus of control from the planner into the executive. Objections to reactivity, though, might rest on the fear that it actually means displacing the locus of control outside of the agent entirely and locating it in the world, leaving the agent to be shoved helplessly around by outside forces. Our solution to this puzzle is to abandon the notion that activity has a particular locus of control at all. Activity arises through interaction, not through control. This transforms our problem from "how do we get this system to control the robot?" to "how do we build this agent to interact with the world in the ways we want?" To answer this question, we need to study the world and we need to find out what sorts of agents can enter into what sorts of interactions with it. In fact, because the agent, the world, and the interactions are interconstraining,

we need to design the agent incrementally, tacking back and forth between deepening our understanding of the world, finding new sorts of interactions with it that might occur, and making our agent more sophisticated so that it participate in these sorts of interactions. We designed Pengi this way. Pengi can't control its world and doesn't try. It is, instead, always ready to participate in various sorts of interactions with bees and ice cubes.

It may be difficult to understand our claim that Pengi uses no plans. In particular, it may seem that Pengi's arbitration network is really a plan; or that the whole central system is; or that whatever state elements Pengi has are plans; or that visualization is really planning. Analogies with some common devices that clearly do not use plans may be helpful.

Some toasters, for example, have photocells that sense when the bread is dark enough. These anticipate the future in the same way Pengi does: they are wired up to act appropriately based on currently perceptible conditions that predict future events. A soda machine, like Pengi, has state elements (forming a coin counter) which are updated based on sensory conditions and their current state, and which determine how the system will behave in the future. An electromechanical elevator controller, like Pengi, engages in complex activity, interacting with other agents whose actions can not be predicted, maintaining state, and using sensors. A well-designed elevator controller, like Pengi, manifests complex, useful dynamics which emerge from its interactions with its environment, rather than being programmed. Yet, we presume, no one would argue that toasters, soda machines, or elevator controllers use plans.

Finally, it is worth mentioning that there is at least one use of the word "plan" current that clearly conforms to neither the plan-as-program view nor the plan-as-communication view. This view, found in some recent work on natural language discourse and on rational action (Grosz and Sidner 1988, Konolige and Pollack 1989), might be called "plans-as-constraints." On this view, plans are sets of beliefs and "intentions," which are constraints on possible futures; the view does not specify how these constraints affect action, but rather how they are used to interpret other agents' actions. (Chapman forthcoming) will discuss the relationship between this view and the other two.

Acknowledgments

In working out these ideas about plans and planning, we were greatly aided by conversations with and comments on drafts from John Batali, Rod Brooks, Gary Drescher, Barbara Grosz, Pattie Maes, Drew McDermott, Tomás Lozano-Pérez, Beth Preston, Jeff Shrager, Penni Sibun, Orca Starbuck, Dan Weld, and Ramin Zabih. Lucy Suchman has greatly influenced our thinking by introducing us to ethnomethodology (Garfinkel 1967, Heritage 1984). This essay itself descends from position papers we prepared for

the DARPA Planning Workshop in Santa Cruz, California in October 1987, and for the COST-13 Workshop On Representation and Learning in an Autonomous Agent in Lagos, Portugal in November 1988. Thanks to Ted Linden and Drew McDermott for organizing the panels where they were presented at Santa Cruz and to Pattie Maes for organizing the Lagos workshop. And finally thanks to Mike Brady, Rod Brooks, and Stan Rosenschein for various sorts of support.

References

Philip E. Agre, Routines, AI Memo 828, MIT Artificial Intelligence Laboratory, 1985.

Philip E. Agre, The dynamic structure of everyday life, PhD Thesis, MIT Department of Electrical Engineering and Computer Science, 1988.

Philip E. Agre and David Chapman, Pengi: An implementation of a theory of activity, Proceedings of AAAI-87.

Philip E. Agre and David Chapman, Indexicality and the binding problem, Proceedings of the AAAI Symposium on Parallel Models, 1988.

King Beach, The role of external mnemonic symbols in acquiring an occupation, in M. M. Gruneberg, P. E. Morris, and R. N. Sykes, eds., *Practical Aspects of Memory: Current Research and Issues*, Volume I, John Wiley and Sons, Chichester, 1988.

Rodney A. Brooks, A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* **2**(1), April 1986, pages 14-23.

Rodney A. Brooks, A robot that walks: Emergent behaviors from a carefully evolved network, AI Memo 1091, MIT Artificial Intelligence Laboratory, 1989.

David Chapman, Planning for conjunctive goals, *Artificial Intelligence* **32**(3), 1987, pages 333-377.

David Chapman, Instruction use in situated activity, MIT Computer Science Department PhD Thesis, forthcoming.

David Chapman and Philip E. Agre 1986, Abstract reasoning as emergent from concrete activity, in M. P. Georgeff and A. L. Lansky (editors), *Reasoning about Actions and Plans*, Proceedings of the 1986 Workshop at Timberline, Oregon, pages 411–424, Morgan Kaufmann, Los Altos CA (1987).

R. T. Chien and S. Weissman, Planning and execution in incompletely specified environments, *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, 1975, pages 169-174.

Jonathan H. Connell, Creature design with the subsumption architecture, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, 1987, pages 1124-1126.

Johan de Kleer, Jon Doyle, Guy L. Steele, Jr., and Gerald Jay Sussman, Explicit control of reasoning, *Proceedings of the ACM Symposium on Artificial Intelligence and Programming Languages*, Rochester, New York, 1977.

Mark Drummond, Situated control rules, *Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems*, University of Rochester, New York, 1989.

R. James Firby, An investigation into reactive planning in complex domains, Proceedings of AAAI-87.

Mark S. Fox and Stephen Smith, The role of intelligent reactive processing in production management, in *13th Meeting and Technical Conference*, CAM-I, November 1984.

Harold Garfinkel, *Studies in Ethnomethodology*, Polity Press, Oxford, 1984. Originally published in 1967.

Michael Georgeff and Amy Lansky, Procedural knowledge, Proceedings of the IEEE, Special Issue on Knowledge Representation, pages 1383-1398, October 1986.

Michael Georgeff and Amy Lansky, Reactive reasoning and planning, Proceedings of AAAI-87, pages 677-682.

Georges Giralt, Raja Chatila, and Marc Vaisset, An integrated navigation and motion control system for autonomous multisensory mobile robots, Proceedings of the First Symposium on Robotics Research, MIT Press, 1984, pages 191-214.

Thomas Gladwin, *East is a Big Bird*, Harvard University Press, 1970.

Barbara J. Grosz and Candace L. Sidner, Plans for discourse, in P. Cohen, J. Morgan, and M. Pollack, eds., *Intentions in Communication*, MIT Press, Cambridge Massachussets, 1988.

Kristian J. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*, Academic Press, 1989.

Caroline Hayes, Using goal interactions to guide planning, Proceedings of AAAI-87, pages 224-228.

Edwin Hutchins, Learning to navigate in context. Manuscript prepared for the Workshop on Context, Cognition, and Activity, Stenengsund, Sweden, August 6–9, 1987. Institute for Cognitive Science, University of California, San Diego, La Jolla, California.

John Heritage, *Garfinkel and Ethnomethodology*, Polity Press, Cambridge, England, 1984.

Kurt Konolige and Martha E. Pollack, Ascribing plans to agents, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989, pages 991-997.

Leslie Pack Kaelbling, An architecture for intelligent reactive systems, in Michael P. Georgeff and Amy L. Lansky, eds, *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop*, Timberline, Oregon, 1986, pages 395-410.

Amy L. Lansky and David S. Fogelsong, Localized representation and planning methods for parallel domains, Proceedings of AAAI-87, pages 240–245.

Jean Lave, *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life*, Cambridge University Press, 1988.

Tomás Lozano-Pérez and Rodney A. Brooks, An approach to automatic robot programming, AI Memo 842, MIT Artificial Intelligence Laboratory, 1985.

Pattie Maes, The dynamics of action selection, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989, pages 991-997.

Chris Malcolm and Tim Smithers, Symbol grounding via a hybrid architecture in an autonomous assembly system, in Pattie Maes, ed., *New Architectures for Autonomous Agents: Task-level Decomposition and Emergent Functionality.* MIT Press, Cambridge, Massachusetts, 1990.

Michael A. Erdmann and Matthew T. Mason, An exploration of sensorless manipulation, *IEEE Journal Robotics and Automation* **4**(4), pages 369-379, 1988.

John McCarthy, The advice taker, reprinted in Marvin Minsky, ed., *Semantic Information Processing*, MIT Press, 1968. Originally published 1958.

Drew McDermott, Planning and Acting, *Cognitive Science* **2**(2), 71–109, 1978.

David P. Miller, Execution monitoring for a mobile robot system, *Proceedings of the SPIE 1989 Conference on Intelligent Control and Adaptive Systems*, Philadelphia, 1989.

George A. Miller, Eugene Galanter, and Karl H. Pribram, *Plans and the Structure of Behavior*, Henry Holt and Company, 1960.

Nils J. Nilsson, Action networks, *Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems*, University of Rochester, New York, 1989.

David W. Payton, Internalized Plans: a representation for action resources, in Pattie Maes, ed., *New Architectures for Autonomous Agents: Task-level Decomposition and Emergent Functionality.* MIT Press, Cambridge, Massachusetts, 1990.

Stanley J. Rosenschein and Leslie Pack Kaelbling, The synthesis of digital machines with provable epistemic properties, in Joseph Halpern, ed, *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, Monterey, California, 1986, pages 83-98.

David E. Rumelhart, Paul Smolensky, James L. McClelland, and Geoffrey E. Hinton, Schemata and sequential thought processes in PDP models, Chapter 14 in James L. McClelland, David E. Rumelhart, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, Massachussets, 1986.

Bob Scher, *The Fear of Cooking*, Houghton-Mifflin, 1984.

Marcel J. Schoppers, Universal plans for reactive robots in unpredictable environments, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, 1987, pages 1039-1046.

Sylvia Scribner, Studying working intelligence, in B. Rogoff and J. Lave, eds, *Everyday Cognition: Its Development in Social Context*, Harvard University Press, 1984.

Herbert A. Simon, *The Sciences of the Artificial*, MIT Press, 1970.

Susan U. Stucky, The situated processing of situated language, CSLI Report 87-80, March 1987.

Lucy Suchman, What is a plan?, ISL Technical Note, Xerox Palo Alto Research Center, 1986.

Lucy Suchman, *Plans and Situated Action*, Cambridge University Press, 1987.

William Swartout, ed, DARPA Santa Cruz Workshop on Planning, *AI Magazine*, Summer 1988, pages 115-131.

Austin Tate, Planning and Condition Monitoring in a FMS, International Conference on Flexible Manufacturing Systems, London, UK, July 1984.

Austin Tate, A review of knowledge-based planning techniques, *The Knowledge Engineering Review* **1**(3), June 1985, pages 4-17.

Shimon Ullman, Visual routines, *Cognition* **18**, 1984, pages 97-159.

Lev S. Vygotsky, *Thought and Language*, MIT Press, Cambridge Massachussets, 1962.

James W. Wertsch, *Vygotsky and the Social Formation of Mind*, Harvard University Press, Cambridge MA, 1985.

Robert Wilensky, *Planning and Understanding: A Computational Approach to Human*

Reasoning, Addison-Wesley, Reading MA, 1983.

David E. Wilkins, Recovering from execution errors in SIPE, SRI Tech Report 346, 1985.

David E. Wilkins, *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kaufmann Publishers, Los Altos CA, 1988.