

In Defense of Reaction Plans as Caches

Marcel J. Schoppers

Ginsberg raises two issues for our consideration: (1) universal plans that do not allow cognitive actions (run-time problem solving) might need an exponential amount of space or circuitry for their realization and (2) universal plans that do allow cognitive actions will spend so much time on these actions that the universal plan itself can be largely superfluous. In sum, depending on whether planning is allowed to supplement a universal plan, the universal plan itself is likely to be either infeasible or superfluous, respectively.

Reaction Plans and Classifiers

Let me clear up some areas of potential confusion. Although universal plans had the title role in Ginsberg's article, he is actually discussing a small group of related endeavors that are more commonly known as reactive

Universal plans address the tension between reasoned behavior and timely response by caching reactions for classes of possible situations. This technique reduces the average time required to select a response at the expense of the space required to store the cache—the classic time-space trade-off. In his article, Matthew Ginsberg argues from the time extreme and against the space extreme. Although I find both extremes undesirable, I defend an increase in space consumption.

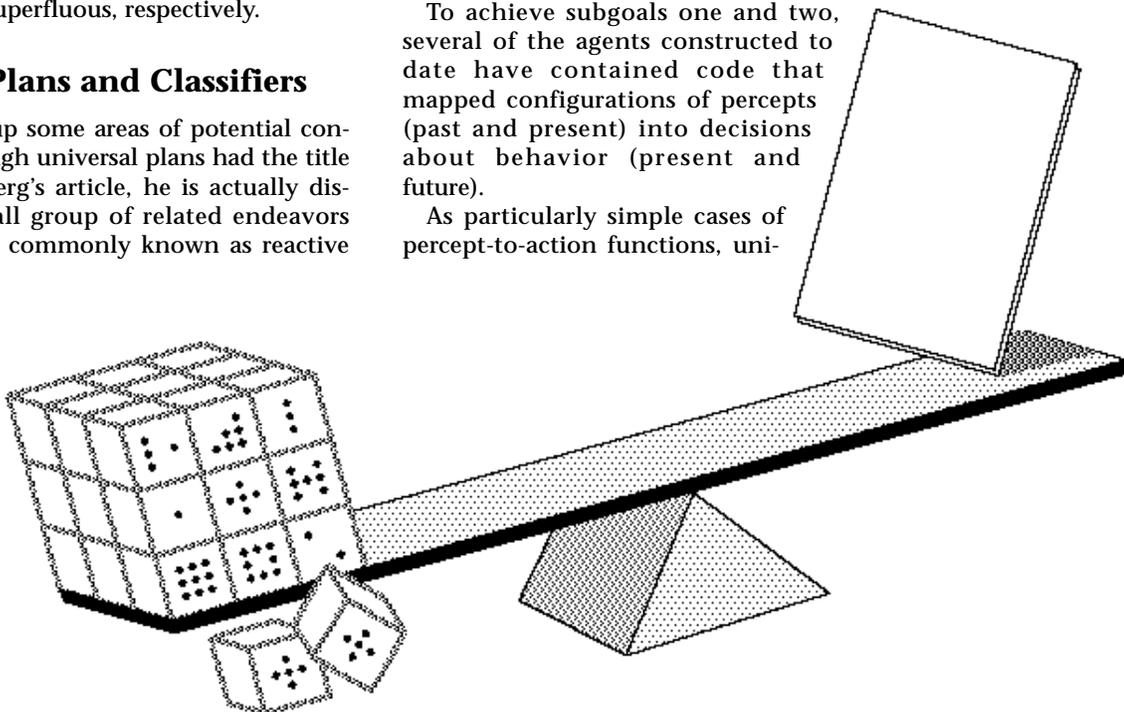
planning and situated action. The universal plans label originally pertained only to my work.

Ginsberg's concerns apply only to a small part of the situated action work. The general

idea is to build embodied agents that behave intelligently in physical surroundings. This goal has a number of subgoals, the most prominent of which are (1) flexibility in interacting with real environments, (2) timeliness of response, and (3) coping with computational and physical limitations.

To achieve subgoals one and two, several of the agents constructed to date have contained code that mapped configurations of percepts (past and present) into decisions about behavior (present and future).

As particularly simple cases of percept-to-action functions, uni-



At issue in this discussion is the potential size of the decision structure that maps percepts into actions.

versal plans map the agent's current percepts into current effector commands. In so doing, universal plans ignore many functions that must be present in real agents, such as robotic control laws to realize primitive actions, perceptual activities to obtain information, recall of past events, the ability to decide what goals to achieve when, a focus of attention, and the ability to act and reason at the same time. These functions are present in the situated agents being constructed by others. Thus, universal plans are a severe oversimplification of the functionality needed in situated agents. However, the oversimplification is deliberate; it makes universal plans amenable to automatic synthesis; and it is by no means my best and final word.

Nevertheless, the simplicity of universal plans has served to clarify certain promising features and potential problems of the situated action work. At issue in this discussion is the potential size of the decision structure that maps percepts into actions. Universal plans make this issue especially prominent by being equivalent to a decision tree whose outcomes are names of effector actions and whose decision nodes are labeled with environmental conditions (whose perception and retention is ignored). A universal plan is executed by repeatedly starting at the top of the decision tree, testing the conditions encountered on the way down, branching right or left as appropriate, then executing the action found at the bottom. Thus, universal plans amount to deeply nested if-then-elses inside a do-forever. (A universal plan is also equivalent to a small set of production rules or to a simple control system.) Because at every test, both outcomes eventually lead to some action, it follows that a universal plan inherently cover every possible world state of the relevant domain. However, this coverage might be trivial, perhaps making only a single distinction and splitting the domain into only two sets of world states.

The plan structure is also capable, though—at least in principle—of distinguishing each possible situation from every other and selecting a unique reaction for every possible world state. This capability is indispensable if the plan structure is to be plausible as a cache

for organizing and indexing reactions. However, if the domain were especially difficult and the universal plan were pushed to completion, the plan might, indeed, become large.

To demonstrate that such functionality could be automatically constructed by a planning program, I had my planner construct the entire plan at once, thereby igniting the plan-size controversy. My purpose was to illustrate that universal plans operate on different principles from plans built automatically: Universal plans classify possible situations; naturally cover the entire domain; place no restrictions on what possible situations might become actual and in what order; and use sensory information to identify the class of possible situations to which the actual situation belongs, hence redeciding what to do from moment to moment.

I have never argued that the entire plan should always be assembled in one sitting; conversely, had I designed the planner to limit itself to the reasoning necessary to solve the current problem, as done with previous planners, the novel aspects of my work would have been much less obvious.¹ Rest assured that the development of a more practical planner is high on my agenda.

However, universal plans have become a focal point for doubts about all the work being done under the situated action heading. Let me repeat that universal plans are severely oversimplified to facilitate their automatic synthesis; there is far more to the situated action enterprise than functions that map perceptual states into actions. Nevertheless, such functions are common in situated action (and control systems) work, so that the current controversy isolates an important part of the approach. Indeed, the idea of classifying perceptual states allows some useful perspectives on the reactive planning field as a whole, for example, by allowing us to distinguish where the reactivity comes from.

On one side, there is work on ways of encoding reactive behavior (Agre 1988; Albus 1985; Allard and Kaemmerer 1987; Andersson 1987; Brooks 1986; Drummond 1989; Firby 1989; Georgeff and Lansky 1987; Giralt, Chatila, and Vaisset 1984; Goldstein and Grimson 1977; Griesmer et al. 1984; Harel 1987; Hendler and Sanborn 1987; Kaelbling 1988; Nilsson 1988; Schoppers 1989; Simmons and Mitchell 1989; Sobek 1985; Weisbin, Saussure, and Kammer 1986). Here, the needed reactivity is available without automatic planning; the programs are generally coded by hand. Some of the researchers doing this work present themselves as doing bottom-up robot programming with the goal

of discovering suitable constructs for a higher-level language. In devising universal plans, I postulated classification (of perceptual states) as one of the essentials of a language for building situated agents and found a way of doing such classification with a data structure that was also manipulable by a planning program. Hence, I describe this work as being about *reaction plans*—plans consisting of simple reactions to possible situations—despite the fact that universal plans are the first programs in this category for which the word plan is really appropriate.

On the other side is work on the problem of planning and replanning when the planner has only a limited amount of time (Chen 1985; Dean and Boddy 1988; Hayes-Roth 1987; Lesser, Pavlin, and Durfee 1988; Masui, McDermott, and Sobel 1983; Ow, Smith, and Thiriez 1988; Wesson 1977). There is a plan, but it is constantly being revised. Here, it is the plan synthesizer that must adapt itself to the pressures of the domain. This work is properly labeled reactive planning.

(This distinction is not a permanent one. A universal plans planner that adjusted its reasoning to take deadlines into account would fall into both categories and might be labeled a reactive reaction planner.)

Ginsberg's misgivings are about the use of reaction plans, most of which contain—in some form—a classifier that maps possible perceptual states into appropriate responses. Therefore, from now on, I talk about classifiers and reaction plans in general and use the name universal plans to mean only my work. To facilitate the rest of this discussion, let me introduce some other terminology:

A *feature* p_i is a Boolean variable, that is, $p_i = \{0, 1\}$. A *situation* is a vector of Boolean values b_1, b_2, \dots, b_n , where b_i denotes the value of the feature p_i so that $b_i \in p_i$. A *condition* is a vector c_1, c_2, \dots, c_n in which some entries are "don't care," that is, $c_i \in p_i \cup \{x\}$. Two conditions are *disjoint* if and only if for some feature, one condition specifies 0, and the other condition specifies 1; otherwise, the two conditions overlap. A *decision rule* is a pair $\langle \text{condition}, \text{outcome} \rangle$. Two rules are *inconsistent* if and only if they overlap and specify different outcomes; otherwise, they are *consistent*. A *classifier* is a set of decision rules that are pairwise consistent. A classifier is *universal* if every possible situation is matched by the condition element of some decision rule.

In an embedded agent, features can correspond to tests on sensory input or tests on previously computed data. The outcomes of decision rules control the agent's actions: Some actions can move effectors, and some

can compute data for later input to features. (In some relevant work, outcomes can be small procedures [Firby 1989; Georgeff and Lansky 1987]). Reaction plans are usually universal.

There are (at least) two ways to construct an agent using classifiers: An outcome can be a subset of the actions available to the agent, so that a single outcome can control several effectors, or there can be many classifiers, one for each action, so that outcomes reduce to on-off signals. The former is more convenient for programming purposes; the latter makes it more obvious how a situated agent's classifier relates to Boolean functions. Unless I indicate otherwise, I assume there is a single classifier whose outcomes indicate sets of actions.

Restatement of Ginsberg's Argument

In referring to cognitive actions, Ginsberg was distinguishing reaction plans that can be supplemented with run-time planning from those which cannot. However, for his argument, the presence or absence of planning is not important. What matters is whether the reaction plan contains an action that can somehow displace and subsume numerous effector actions (as might happen if this action were called PLAN); if so, the reaction plan can be made smaller than it would be if the displaced effector actions were an explicit part of the plan. Thus, Ginsberg would be better served by distinguishing implicit action selection from explicit action selection, as follows: A reaction plan is *explicit* if effector actions cannot be performed except by being named in the outcome of the active decision rule.

Rather than talking about plans containing cognitive actions, I talk about plans that can select actions implicitly (the latter are a subset of the former). In so doing, I am strengthening Ginsberg's argument because permitting cognitive actions enlarges both the number of actions and the number of sensors.

There is also the issue of whether the situated agent's classifier is exact or underfitted. That is, given that the classifier is universal, does it always select a suitable reaction (effector or cognitive) for every possible situation, or does it fail to properly discriminate some situations, thus assigning some situations an incorrect or poor choice of action(s) but winning a reduction in the complexity of the classifier?

Ginsberg's argument is mainly with explicit reaction plans, even if they are underfitted as classifiers. For such reaction plans, it is easy to show that a hardware implementation might

. . . Ginsberg would be better served by distinguishing implicit action selection from explicit action selection . . .

require exponentially large numbers of gates. However, Ginsberg's calculation is incomplete, answering the question, "How many gates does it take to ensure that it is possible to build any one of the possible reaction plans from s sensors to o actions?" or, more succinctly, "How many gates are needed to build the worst possible Boolean function having s inputs and o outputs?" This question is rather different than the more appropriate one: "Given a fixed (randomly selected) Boolean function having s inputs and a outputs, how many gates does it need?" Although Ginsberg did not make the connection clear, the argument can be salvaged. Indeed, a sharp version of the same argument was first advanced by Claude Shannon (1949), whose argument applied to Boolean functions with only one output, roughly as follows:

- There are 2^{2^s} Boolean functions of s variables.
- With g binary gates, it is possible to build at most $(16(g + s + 2)^2)^g$ Boolean functions on s input.
- If we happen to set $g = 2^s / 4s$, the above bound is $((2^s/s)^2)^g = 2^{2sg} / s^{2g} = 2^{(2^s/2)} / s^{2g}$, which is much smaller than 2^{2^s} ; that is, even with an exponential number of gates, you can build circuits for only a small fraction of the possible Boolean functions.
- Because each circuit computes only one function, the vast majority of Boolean functions require circuits of size larger than $2^s / 4s$.

You can make this argument apply to reaction plans by merely regarding them as a collection of o Boolean functions of s input each.

Ginsberg then anticipates two answers: First, reaction plans are not randomly selected classifiers; that is, they can be simpler than the average case. Second, reaction plans can be underfitted classifiers, so that again they can be simpler than the average case.

Ginsberg counters both answers by pushing up the number of sensors and, hence, strives to remain pessimistic about the utility of all explicit reaction plans, whether or not they are underfitted.

Finally, Ginsberg considers reaction plans that initiate planning, perhaps by means of a cognitive action called *PLAN*.² For these reaction plans, Ginsberg's argument changes; now, he expects that the vast majority of the agent's time will be spent on this planning, so that the effector actions—and the reaction plan itself and, indeed, the whole reaction plan enterprise—are in danger of being superfluous and irrelevant.

Nine Possible Answers

There are many answers to Ginsberg's arguments. The first group of answers addresses the gate complexity argument and shows it to be misguided and, probably, moot.

Answer 1

Consider any arbitrary program that does not reason about its own activity. The vast majority of programs are of this kind, including many expert systems. Production systems are an especially good example because, like reaction plans, they map a large number of features (ground literals potentially in working memory) into a response (some set of rules to be fired). Ginsberg's argument transfers as the conclusion that production systems might need a number of rules which is exponential in the number of features (possible ground literals); therefore, production systems—and indeed, all nonplanning programs—are a bad idea. Because there are many useful nonplanning programs in the world, it is obvious that Ginsberg has overlooked something: variables, which allow a fixed program to respond to arbitrarily large amounts of data. Variables (of an indexical-functional sort) allow the *PENGI* program (Agre and Chapman 1987) to cope with a world containing any number of bees and ice blocks. In addition, variables support recursion, by which means a universal plan capable of building a tower of three blocks can also build towers of arbitrary height (Schoppers 1989) (compare Waldinger's recursive plans [Manna and Waldinger 1987]). Many builders of reaction plans allow the use of variables in their situated agents. The run-time use of variables causes the domain-size problem to manifest itself not as a plan-size problem but as a planning-time problem.

Answer 2

This answer and the next comprise an informal argument to the effect that even the average Boolean function is well beyond the execution capabilities of our (unaided) human brain-machine. That is, because our brains seem quite unable by themselves to use even mildly complex Boolean functions, robots shouldn't need them either, and fears of exponential gate complexities are premature.

Let us momentarily adopt the view that each action comes with its own classifier which decides on the appropriateness of this individual action; the situated agent contains o such classifiers, each of which is a Boolean function of the s sensors.³ It is a startling fact

. . . human cognition is fundamentally limited to conceiving and constructing reaction plans . . . which grow at most linearly in the number of relevant inputs.

that computational complexity theorists have been unable (to date, after about 20 years of concerted effort) to identify any Boolean function whose realization in hardware requires a number of binary gates which grows worse than linearly with the number of inputs (Redkin 1970; Blum 1984; Boppana and Sipser 1988; Wegener 1987).⁴ The most complex functions identified to date are parity and threshold functions, for which lower bounds of $3N$ or $4N$ gates have been proved; the $4N$ result was obtained by restricting the gate types to AND, OR, and NOT. Whatever the reason behind the lack of progress by complexity theorists, their continued perplexity is one argument that humans have no pressing need for Boolean functions with non-linear gate complexities.

Answer 3

Against the previous answer, it can be argued that if reaction plans are taken as objectively given by a domain and not somehow mathematically convenient or natural to humans, then the gate complexity of reaction plans might be exponential after all. However, this argument fares badly in light of results from the field of automatic concept induction. For some chess end games, it is possible to construct a complete table of legal positions and associate each position with the won-lost-drawn outcome that would be obtained under optimal play. This table can then be fed through an automatic induction algorithm such as Quinlan's (1983) ID3, which will come out with a decision tree that classifies every position of the end game. This decision tree might be regarded as a reaction plan for the end game: It tests for the presence (or absence) of certain positional features and then indicates whether the appropriate response is to play on (won), resign (lost), or offer a draw.

There is no reason to expect that such a decision tree will be easily interpretable by human beings because it arises from the rules of the game through a potentially lengthy lookahead. In fact, in one experiment, the automatically induced decision tree was not only quite unintelligible to humans but ran about five times faster than the best equiva-

lent program that could be constructed by the experimenter, despite his having considerable programming skills and subsequently spending several months trying to catch up (Michie 1983; Quinlan 1982). Thus, the chess end-game arena contains reaction plans that are not at all natural to human cognition.

Nevertheless, the use of interactive structured induction techniques has allowed the synthesis of intelligible decision trees for such cognitively unfriendly end games (Shapiro and Niblett 1982; Shapiro 1983; Michie 1982). These decision trees look quite different from the one that enjoyed the fivefold speed advantage, however. In fact, these decision trees are linear; that is, they can be realized with a number of gates which is roughly equal to the number of relevant positional features. Thus, even when a classifier is objectively so foreign to human cognition that we cannot approximate it for trying, the function encoded by this classifier can still be learned and used by humans; however, this learning only takes place at the expense of selecting new positional features and constructing a different classifier that is of approximately linear size in the number of features. This evidence further strengthens the argument that human cognition is fundamentally limited to conceiving and constructing reaction plans (or other classifiers) which grow at most linearly in the number of relevant inputs. Unless robots must be inherently more complex than people, fears of exponential gate complexities are currently unfounded.

Answer 4

When considering the possibility that many useful functions might be encodable in less than an exponential amount of circuitry, Ginsberg responds by enlarging the number of relevant problem features (for example, by introducing more blocks and more factors in deciding how to control the baby). However, clearly, such an enlargement is not a serious threat if circuit size (or space consumption) is only a small polynomial in the number of features. Indeed, the fact that humans are capable of reasoning with surprisingly large numbers of domain-dependent features is yet further evidence that we need not be con-

The existence of routine behavior is the foundation on which the situated action work rests.

cerned with either the number of features or the consequent space consumption; there are psychological data on this issue, again in the domain of chess. Master-level players are capable of reconstructing a single unseen, randomly selected (but naturally occurring) chess position by asking about 70 yes-no questions on average (Nievergelt 1977). Because the positional features needed to complete a given position can depend on the features already known, it follows that the total number of features available for use by master-level players is probably many times 70. Even so, a mere 70 features—if they are independent—already allow for $2^{70} \cong 10^{21}$ possible positions; and it has been estimated that only about 10^{15} positions occur in practice (Jongman 1968).

Thus, it appears that a reaction plan for chess is not as implausible as one might think. Indeed, such a reaction plan is not only theoretically within the bounds of human capability, but there is some evidence that it is already being constructed in the heads of chess masters. It has been estimated that masters use a repertoire of about 30,000 features to classify and identify positions (Simon and Gilmartin 1973).⁵ Using these features, masters can both encode and remember a randomly selected, plausible position after studying it for only about five seconds (Chase and Simon 1973; DeGroot 1965).⁶ In five seconds, amateurs observe and remember little, but masters are nearly flawless, suggesting that the masters have developed most of the identification and recognition capabilities needed for a reaction plan for chess (as they experience it).⁷

Answer 5

To the best of my knowledge, neither I nor any of my colleagues has vowed to limit ourselves to explicit reaction plans. Hence, the gate complexity issue is probably moot.

Now I turn to Ginsberg's second argument; namely, if a reaction plan is supplemented with planning, the planning will be so predominant that the reaction plan itself is barely useful.

Answer 6

With Ginsberg's argument applied to ordinary programs once again, suppose you have a program that is capable of both retrieving some precomputed results and computing new ones. Ginsberg's expectation is that if such a program is allowed to compute new results, then such computing will so predominate that the precomputed results will seldom be needed. However, the program I just described is common; it is a program that has built up a cache of previous responses. Although there are certainly some general issues about what computations are worth caching, Ginsberg seems to be asking for evidence that caching could ever be helpful. Alternatively, he considers that in real life, activities recur so infrequently that there is never any reason to cache a reaction. This expectation flies in the face of common experience, which can be so routine that it becomes boring. How many cups of tea (or coffee) does a person make in a lifetime? After some practice, the tea can be made even while the tea maker is thinking of some other event or fact. The existence of routine behavior is the foundation on which the situated action work rests.

Answer 7

Remember that when faced with a persistent tower-destroying baby, a reaction plan for building block towers might continue trying to build the block tower, no matter how long the baby continued to interfere. This problem has been dubbed futile looping (Firby 1989). Ginsberg used this scenario as an argument that reaction plans might need large numbers of features; however, I argue that the flexibility Ginsberg seems to demand is more easily obtained from reaction plans than from a planner. Suppose we were to replace the reaction plan with a classical planner and execution monitor. Such a system would respond to the baby's destructive tendencies by revising the plan, which would then try to complete the tower; of course, though, the baby would interfere again, and the planner would be forced to replan again, ad infinitum. I know of no planning system capable of either recognizing repeated interference or doing something about it.⁸ Persistent interference is a problem for present-day planners and reaction plans alike; it is merely more obvious in reaction plans because they can recover without replanning—and, incidentally, without the comparatively horrendous replanning costs of a classical planner. Indeed, in reaction plans, it is possible to detect interfer-

ence, respond to it on (say) the third occurrence, and then try something else (Firby 1989); so, planners are somewhat behind on this issue.

Even if I were inclined to give planners the benefit of the doubt—suppose it was easy for a planner to recognize and terminate repeated interference—the problem does not end here. Whether the baby is our own, she or he will probably be back to randomize our abode again. Does it seem reasonable that the planner should have to build the same piece of plan (for terminating interference) many times over? The issue is forced by the fact that an important difference exists between a situated agent and a planner: Real life is not over when the problem is solved. Clearly, we come to a choice between enlarging a reaction plan once or planning the same counterinterference steps many times—this is the classic time-space trade-off. The situated action community prefers a middle ground.

Answer 8

In arguing that reaction plans should be defended as a compaction technique, Ginsberg overlooked the time savings afforded by a precomputed classifier. Nevertheless, even without such time savings, reaction plans would be well worth researching because they significantly reduce the space required to cache plans, as I now show.

In classical planning, the domain is conceived as a state-space graph, a (linear) plan is a path through the state-space graph from a start node to a goal node, and planning is a heuristic search for such a path. Let me refer to such plans as *path plans*. Finding a path from every possible start node to a goal node—which is the competence of a reaction plan—requires that the planner traverse every node of the state-space graph at least once. Therefore, if the domain permits N possible states, a naive approach to caching requires space of $O(N)$. (For the sake of a simple analysis for both path plans and reaction plans, I disallow such space-saving tricks as variablizing, along with such use of abstraction as would require searching to reconstruct a complete plan; under these conditions, path planners cannot cover and cache the entire domain in less than $O(N)$ space.) The task of a reaction plan planner is not to traverse the state-space graph to find paths but to partition it into groups of states such that all states in a given group require the same reaction. The expected cardinality of a randomly selected partition on N objects is $O(N/\log N)$ (Haigh 1972). That is, the average reaction plan for a

domain containing N world states has $O(N/\log N)$ decision rules. This is a marginal improvement; a domain of 10^{12} states would require 4×10^{10} decision rules. However, now it turns out that reaction plans can exploit abstraction in a way unavailable to path plans. The caching of abstractions of path plans only saves space if the most detailed abstraction levels can be omitted from the cache. Such caching would require searching to recover a complete plan, and I just outlawed such computation. When using abstract reaction plans, however, the domain can be partitioned with the most abstract features, then each of these partitions can be partitioned again and again, down to the bottom of the abstraction hierarchy. A reaction plan with abstraction hierarchies will (on average) partition N states into only $O(N^{2/3})$ groups of states.⁹ This savings is more significant: It means that a domain of 10^{12} states can be covered with 10^8 decision rules. Thus, reaction plans can require only 0.01 percent of the space required to cache path plans of equivalent competence. (Note: This result must be qualified by the possibility that some storage reduction techniques might be more effective with path plans than with reaction plans.)

Answer 9

To convince Ginsberg that the compaction afforded by reaction plans is not only numerically significant but also technologically significant, I must convince him there will be many domains in which reaction plans can be used and in each such domain many reactions worth caching.¹⁰ Answer 8 suggests that reaction plans should be considered whenever reactions are worth caching, even if there are only a few. Whether it is worthwhile to cache any particular reaction is an open question, but caching is more likely to be worthwhile for agents that must repeatedly achieve the same goal from many starting conditions.¹¹ This criterion is met when (1) the domain is so dynamic that the agent can encounter many world states in the course of a single attempt on the goal and (2) when the domain is just dynamic enough to make the initial state vary from one attempt to the next.

An example of the first point is driving; an example of the second is making a cup of tea. Situated agents are more likely than most to encounter both kinds of tasks; the ubiquity of routine is eloquently described by Agre (1988). To put it more strongly, even if answer 8 were ignored completely—that is, if reaction plans afforded no compaction whatsoever—then reaction plans should still be researched for

. . . I must convince him there will be many domains in which reaction plans can be used and in each such domain many reactions worth caching . . .

their ability to reduce total planning time, the survival value of a ready response, and the potential to automate the construction of knowledge-based reactive systems.¹²

Note that I am not, as Ginsberg suggests, “making an extremely strong claim—that the planning problems encountered by autonomous agents will be of the type for which universal plans can be represented.” I clearly do expect that some parts of some domains can be solved with reaction plans, but I am willing to let the applicability of reaction plans be an empirical issue.

Meanwhile, I expect the situated action enterprise to produce (at least) the following new abilities:

1. Formal analysis of the information content of embedded systems, with a view to providing guarantees on the reasonableness and safety of the behavior that might result (Rosenschein and Kaelbling 1985, 1988)
2. Run-time problem solving, which might significantly extend system competence and complexity (still subject to the just-mentioned guarantees) and might converge with work on restructurable control systems (systems that can compensate for malfunctioning effectors [Montoya 1982])
3. Techniques for the automatic construction of parts of rule-based and embedded systems (Schoppers 1989) (when done in parallel with plan execution, such construction supports automatic and incremental acquisition of new competence)
4. Automatic refinement of domain models (that is, improving control reliability by learning about the plant being controlled [Carbonell and Gil 1987; Kadie 1988; Mason, Christiansen, and Mitchell 1989]), with subsequent revision of the system’s behavior.¹³
5. Interaction with human supervisors concerning current control constraints, policies, and intentions and the reasoning behind them

Acknowledgments

My special thanks to Peter Ladkin for being available as a guide into the literature on the complexity of Boolean functions. Thanks also to Jonathan Litt for a bibliography on restructurable control, Donald Mitchell for a bibliography on chess expertise, and Ed Reingold for a bibliography on Bell and Catalan numbers. Phil Agre, David Chapman, James Firby, Erann Gat, Leslie Kaelbling, Ted Linden, Tom Mitchell, Stan Rosenschein, Stuart Russell, and the members of the Principia group at Stanford University have all contributed through their helpful conversations.

Notes

1. In Schoppers (1987), I give some arguments for why building an entire plan might not be as bad as it first appears.
2. Reaction plans containing plan actions are not quite the complement of explicit reaction plans because reaction plans can be explicit and initiate planning; however, it is assumed that implicit reaction plans must resort to planning.
3. A Boolean function has only one binary output signal, so multiplication circuits (for example) are excluded; however, circuits that test whether one number is a product of two others are allowed.
4. Because the Boolean function must be defined for any number of inputs, the function is actually an infinite set of functions, but the description of the function must obviously remain of finite size.
5. In theory, 134 features suffice to distinguish each of the 10^{40} legal chess positions. Why masters learn to recognize such an abundance of theoretically superfluous features is explained by answer 3.
6. Five seconds is too short to be creating new features on the fly, which is evidenced by the fact that when they are reconstructing implausible positions, grand masters perform as poorly as amateurs.
7. Lest the reader wonder why anyone would bother to construct a reaction plan for chess, remember that masters become masters only by devoting their lives to the game.
8. That is, I know of no planner capable of noticing that its plan has failed not once but several times over.
9. In a draft of this article, I included appendixes in which I showed that the expected cardinality $c(N)^{c(N)} < e^{(N-1)}$ and derived the $N^{2/3}$ result. Copies of this draft are available from the librarian at Advanced Decision Systems.
10. A cached reaction is significantly more useful than a cached plan because a reaction can be used independently of what happened just before or after it.
11. Michie (1977) measures the computational utility of a whole cache, and Barnett (1984) measures the value of particular pieces of advice, but both approaches ignore the urgency of the problem situation and the frequency of use of the cached advice. Natarajan and Tadepalli (1988) give sufficient conditions on the possibility (not the utility) of learning for performance improvement.
12. In arguing that the reaction plans approach is in trouble even if planning costs are ignored, Ginsberg is thereby ignoring the very problem on which reaction plans are an assault.
13. Although points 2, 3, and 4 might change the system’s behavior, they differ as follows: Restructurable control (point 2) requires no

dynamic extension of the reaction plan (point 3) nor learning about the domain (point 4). Dynamic extension of the plan (point 3) might happen for reasons of plan completeness, but learning about the domain (point 4) implies that even a complete plan might need revising.

References

- Agre, P. 1988. *The Dynamic Structure of Everyday Life*. Ph.D. diss., AI Lab, Massachusetts Institute of Technology.
- Agre, P., and Chapman, D. 1987. Pengi: An Implementation of a Theory of Activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 268–272. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Albus, J. 1985. *Brains, Behavior, and Robotics*. Chichester, England: Byte.
- Allard, J., and Kaemmerer, W. 1987. The Goal/Subgoal Knowledge Representation for Real-Time Process Monitoring. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 394–398. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Andersson, R. 1987. Real-Time Expert System to Control a Robot Ping-Pong Player. Ph.D. diss., Dept. of Computer Science, University of Penn.
- Barnett, J. 1984. How Much Is Control Knowledge Worth? A Primitive Example. *Artificial Intelligence* 22:77–89.
- Blum, N. 1984. A Boolean Function Requiring $3n$ Network Size. *Theoretical Computer Science* 28: 337–345.
- Boppana, R. and Sipser, M. 1988. The Complexity of Finite Functions, Technical Report, Dept. of Mathematics, Massachusetts Institute of Technology.
- Brooks, R. A. 1986. A Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1): 14–23.
- Carbonell, J. and Gil, Y. 1987. Learning by Experimentation. In *Proceedings of the Fourth International Workshop on Machine Learning*, 256–266. San Mateo, Calif.: Morgan Kaufmann.
- Chase, W. and Simon, H. 1973. Perception in Chess. *Cognitive Psychology* 4:55–81.
- Chen, D. 1985. Shallow Planning and Recovery Planning Based on the Vertical Decomposition of the Flight Domain. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1064–1066. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.
- Dean, T. L., and Boddy, M. 1988. An Analysis of Time-Dependent Planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–5, 103–113. Menlo Park, Calif.: American Association for Artificial Intelligence.
- DeGroot, A. 1965. *Thought and Choice in Chess*. The Hague, Holland: Mouton.
- Drummond, M. E. 1989. Situated Control Rules. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, Calif.: Morgan Kaufmann.
- Firby, R. J. 1989. Adaptive Execution in Complex Dynamic Worlds. Ph.D. diss., RR-672, Dept. of Computer Science, Yale Univ.
- Georgeff, M. P., and Lansky, A. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 677–682. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Giralt, G.; Chatila, R.; and Vaisset, M. 1984. An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots. In *Proceedings of Robotics Research, the First International Symposium*, 191–214.
- Goldstein, I., and Grimson, E. 1977. Annotated Production Systems: A Model for Skill Acquisition. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 311–317. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc.
- Griesmer, J., Hong, S. J.; Karnaugh, M.; Kastner, J. K.; Schor, M. I.; Ennis, R. L.; Klein, D. A.; Milliken, K. R.; and Van Woerkom, H. M. 1984. YES/MVS: A Continuous Real-Time Expert System. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, 130–136. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Haigh, J. 1972. Random Equivalence Relations. *Journal of Combinatorial Theory (A)* 13:287–295.
- Harel, D. 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* 8:231–274.
- Hayes-Roth, B. 1987. Dynamic Control Planning in Adaptive Intelligent Systems. In *Proceedings of the Defense Advanced Research Projects Agency Knowledge-Based Planning Workshop*, 4-1–4-7. Arlington, Va.: Science Applications International Corp.
- Hendler, J. and Sanborn, J. 1987. A Model of Reaction for Planning in Dynamic Environments. In *Proceedings of the Defense Advanced Research Projects Agency Knowledge-Based Planning Workshop*, 24-1–24-10. Arlington, Va.: Science Applications International Corp. 24-1ff.
- Jongman, R. 1968. *Het Oog van de Meester* (“The Eye of the Master”). Amsterdam: Van Gorcum.
- Kadie, C. 1988. Diffy-S: Learning Robot Operator Schemata from Examples. In *Proceedings of the Fifth International Workshop on Machine Learning*, 430ff.
- Kaelbling, L. 1988. Goals as Parallel Program Specifications. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 60–65. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Lesser, V.; Pavlin, J.; and Durfee, E. 1988. Approximate Processing in Real-Time Problem Solving. *AI Magazine* 9(1): 49–61.
- Manna, Z., and Waldinger, R. 1987. How to Clear a Block: A Theory of Plans. *Journal of Automated Reasoning* 3(4): 343–377.
- Mason, M.; Christiansen, A.; and Mitchell, T. 1989. Experiments in Robot Learning. In *Proceedings of*

- the Sixth International Workshop on Machine Learning, 141-145. San Mateo, Calif.: Morgan Kaufmann.
- Masui, S.; McDermott, J.; and Sobel, A. 1983. Decision Making in Time-Critical Situations. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 233-235. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.
- Michie, D. 1982. Experiments on the Mechanization of Game Learning: 2. Rule-Based Learning and the Human Window. *BCS Computer Journal* 25:105-113.
- Michie, D. 1977. A Theory of Advice. In *Machine Intelligence* 8:151-168. Chichester, England: Ellis Horwood.
- Michie, D. 1983. Inductive Rule Generation in the Context of the Fifth Generation. In Proceedings of the Fourth International Workshop on Machine Learning, 65-70. San Mateo, Calif.: Morgan Kaufmann.
- Montoya, R., et al. 1982. Restructurable Controls, Technical Report NASA CP-2277, NASA Langley Research Center.
- Natarajan, B., and Tadepalli, P. 1988. Two New Frameworks for Learning. In Proceedings of the Fifth International Workshop on Machine Learning, 402-415. San Mateo, Calif.: Morgan Kaufmann.
- Nievergelt, J. 1977. Information Content of Chess Positions. *SIGART Newsletter* 62:13-15.
- Nilsson, N. J. 1988. Action Networks. In Proceedings of the Rochester Planning Workshop, 21. Rochester, N.Y.: University of Rochester.
- Ow, P. S.; Smith S.; and Thiriez, A. 1988. Reactive Plan Revision. In Proceedings of the Seventh National Conference on Artificial Intelligence, 77-82. Menlo Park, Calif.: American Association for Artificial Intelligence.
- Quinlan, J. R. 1983. Learning Efficient Classification Procedures and Their Application to Chess Endgames. In *Machine Learning: an Artificial Intelligence Approach*, 463-482. Palo Alto, Calif.: Tioga.
- Quinlan, J. R. 1982. Semi-Autonomous Acquisition of Pattern-Based Knowledge. In *Introductory Readings in Expert Systems*, ed. D. Michie, 192-207. New York: Gordon and Breach.
- Redkin, X. 1970. Complexity of Realization of Incompletely Defined Boolean Functions. *Automation and Remote Control* 3: 1474-1477.
- Rosenschein, S. 1985. Formal Theories of Knowledge in AI and Robotics. *New Generation Computing* 3(4): 345-357.
- Rosenschein, S., and Kaelbling, L. 1988. The Synthesis of Digital Machines with Provable Epistemic Properties. In Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge, 83-98. San Mateo, Calif.: Morgan Kaufmann.
- Schoppers, M. J. 1989. Representation and Automatic Synthesis of Reaction Plans. Ph.D. diss., Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign.
- Schoppers, M. J. 1987. Universal Plans for Reactive Robots in Unpredictable Environments. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 852-859. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.
- Shannon, C. 1949. The Synthesis of Two-Terminal Switching Circuits. *Bell Systems Technical Journal* 28(1): 59-98.
- Shapiro, A. 1983. The Role of Structured Induction in Expert Systems. Ph.D. diss., Machine Intelligence Research Unit, Univ. of Edinburgh.
- Shapiro, A., and Niblett, T. 1982. Automatic Induction of Classification Rules for a Chess Endgame. In *Advances in Computer Chess* 3, ed. M. R. B. Clarke, 73-91. Oxford: Pergamon.
- Simmons, R., and Mitchell, T. 1989. A Task Control Architecture for Mobile Robots. In Proceedings of the AAAI Spring Symposium on Robot Navigation, 85-89. Stanford, Calif.: Stanford University.
- Simon, H., and Gilmarin, K. 1973. A Simulation of Memory for Chess Positions. *Cognitive Psychology* 5:29.
- Sobek, R.P. 1985. A Robot Planning Structure Using Production Rules. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1103-1105. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.
- Wegener, I. 1987. *The Complexity of Boolean Functions*. New York: Wiley.
- Weisbin C.; Saussure, G.; and Kammer, D. 1986. A Real-Time Expert System for Control of an Autonomous Mobile Robot Including Diagnosis of Unexpected Occurrences. In Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference 729 (Space Station Automation II), 49-58.
- Wesson, R. 1977. Planning in the World of the Air Traffic Controller. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 473. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.



Marcel Schoppers obtained his B.Sc. in 1979 from the Australian National University in Canberra, Australia. After a year as a systems programmer for the Australian Government Public Service, he returned to the Australian National University and obtained his M.Sc. in 1982. He then moved to the United States to pursue a Ph.D. degree at the University of Illinois. In 1985, he joined SRI for one year, during which he designed and implemented reactive navigation software for the SRI AI Center's mobile robot. Since 1986, he has worked for Advanced Decision Systems. He obtained his Ph.D. from the University of Illinois in 1989 for his work on universal plans and intends to pursue a research career in real-time AI or knowledge-based control systems.