A Perspective on Multiagent Planning

Technical Note 474

September 25, 1989

By: Amy L. Lansky
Artificial Intelligence Center
Computer and Information Sciences Division
and
Center for the Study of Language and Information
Stanford University

APPROVED FOR PUBLIC RELEASE:
DISTRIBUTION UNLIMITED

This research has been made possible in part by the National Science Foundation under Grant IRI-8715972 and the National Aeronautics and Space Administration under Contract NCC2-494. The views and conclusions contained in this paper are those of the author and should not be interpreted as representative of the official policies, either expressed or implied, of NSF, NASA, or the United States Government.

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

ABSTRACT

This report presents an informal review of current research trends in planning and, in particular, my own views on multiagent planning. A short description of the GEMPLAN research project is provided, including both the current state of the system and my future research plans.

1 Multiagent Planning

The primary motivation underlying all of my work in planning has been to handle the complexities and special properties of domains with inherent parallelism – i.e., multiagent domains. Given that humans are notoriously bad at creating coordinated multiagent plans, automated planners for such tasks are clearly necessary. Applications are innumerable. They range from scheduling of already generated plans (for example, creating a schedule for a factory floor where the processes are already well defined) to actual creation of coordinated plans to meet goals (for example, generating a plan for constructing a building that properly coordinates contractors and in which the actual construction steps are not given a priori – they are generated from underlying domain knowledge) to dynamic (execution time) creation and/or modification of coordinated plans for multiple agents (for example, coordinating multiple rovers on the moon as they go about their tasks).

The specific focus of my work has been the development of the GEM multiagent domain model and the GEMPLAN planner based on this model [10, 11, 12, 13, 14]. This work will be described in more detail in Section 3.

Because of the unique properties of multiagent domains, many traditional approaches to planning are inadequate. For example, traditional planning formalisms are not sufficiently expressive. Traditional planners are unable to scale-up to face the formidable complexity challenges of multiagent domains. Like other forms of planning, multiagent planning and coordination must also deal with the problem of reasoning in dynamic environments. However, the task of coordinating multiple agents has special needs and characteristics that make otherwise suitable new planning techniques (e.g., situated actions or other reactive techniques) less suitable. We describe several of these multiagent domain characteristics below.

1.1 Correctness

Probably the most essential and difficult problem faced by multiagent planners is maintaining correctness in the face of conflict. There are two aspects to this problem: its criticality and its inherent costliness and complexity.

When several agents are operating in the world and can potentially conflict, avoiding dangerous situations becomes a critical priority. The philosophy behind reactive approaches – essentially, to react as the situation warrants and, hopefully, to get out of dangerous situations reactively as well – may be suitable for single agent problems or multiagent domains in which conflicts are minimal, relatively unimportant, or avoidable. However, such approaches are too shortsighted to be adequate for full multiagent coordination.[1] In general, more complete plans of action must be analyzed in order to assure correct behavior.

This can be seen by examining how people function in the world. We do not need to plan in advance when our actions do not interfere with others or when we can "work around" any potential problems that might arise. However, when the actions of many people must be coordinated (e.g., setting up a meeting, scheduling a factory, setting up a plan of action for a large organization), advanced planning becomes the norm. There are at least two reasons for this:

1. Interagent conflicts tend to propagate and may therefore affect the entire plan of action for each agent. Once execution begins, it might be too late to change agent plans adequately.

2. Interagent conflicts may have disastrous consequences. Imagine the case of an airplane that runs out of fuel because it did not anticipate landing conflicts and delays caused by other aircraft.

As far as complexity goes, it is well known that enforcing correct conflict-free behavior in the face of parallel activity is, for most forms of conflict, at least NP-hard.

Of course, not all multiagent planning can or should be done in advance. For all the reasons that have been discussed repeatedly in the recent planning literature, some domains are inherently dynamic in nature. Some decision making and planning must be done during execution. The trick is to maintain correctness as much as possible in the face of this. The approach that I have recently been exploring is the use of *run-time "synchronizers"* or constraint satisfaction mechanisms. Examples of this from everyday life include traffic lights, lines in stores, and dispatch rules on the factory floor. Such mechanisms are also the most common means of coordination and resource allocation used by operating systems (for example, priority queues for controlling access to processor cycles). The distinction between run-time synchronizers and the reactive methodologies recently advocated by others is that synchronizers can have well-defined semantic effects on plans. Thus, they can be integrated into plans and reasoned about in advance, as well as used during execution. I will discuss this more in Section 3.

[1]Recent work by Drummond on situated control rules [4], however, is one attempt to reason about the possible effects of reactive techniques and to limit possible executions based on desired effects. However, this approach cannot *creatively generate* new actions or reactions – and therefore has somewhat limited capability for true coordination.

1.2 Higher Level of Reasoning

Most interagent coordinative problems are high level in nature. That is, they deal with actions at a relatively high grain of detail. For instance, a robot wouldn't need (and probably shouldn't even try) to coordinate its low-level motor actions with those of another robot. This is true partly because low-level actions tend not to interfere. But it is also due partly to necessity – coordinative reasoning on a low level of detail would be highly intractable.

Because coordinative reasoning tends to be high level in nature, it is less subject to the dynamic pressures that characteristically appear in domains that involve low-level control. It is simply natural that low-level phenomena change more quickly than high-level phenomena. For example, low-level motor control for robots is often strongly influenced by the changing state of the environment. Thus, situated or reactive approaches make most sense. In contrast, much high-level coordinative reasoning can safely be done in advance. This lends support to the notion of a reasoning architecture (for coordination) that extends *from* preplanning *toward* dynamic/reactive reasoning, rather than trying to start with low-level reactive mechanisms and extending them to deal with higher-level coordination problems. As discussed earlier, reactive mechanisms also do not have adequate temporal context to correctly synchronize agents in the fullest sense.

1.3 Expressiveness

It has been repeatedly established that domain modeling techniques based on the situation calculus (typically, STRIPS-based rules [6]) are inadequate for expressing the complex domain requirements of multiagent domains. Most obvious is their inability to express desired forms of simultaneity (e.g., cooperation) and the potentially desirable consequences of performing interacting actions simultaneously rather than sequentially [12]. Representations such as the interval calculus [2] and my own event-based GEM representation are attempts to attain expressive adequacy. In GEM, a wide variety of properties can be naturally described, including priority requirements, simultaneity requirements, desired patterns of behavior expressed as regular expressions, and interval constraints, as well as the more traditional STRIPS-based properties.

Despite the obvious need to grow beyond STRIPS-based descriptions in order to handle multiagent domains, the AI community has been slow to meet this challenge. Most systems still assume that all potentially concurrent actions do not interact in any way. GEMPLAN is one of the few planners that faces this problem head on.

1.4 Locality: Partitioned Reasoning and Scaling Up

Another obvious requirement of multiagent domains[2] – from both a representational point of view and an efficiency point of view – is the need to explicitly *partition* a domain according to its naturally occurring structures. For example, both agents and resources should be represented as regions or *localities* of activity, each of which has its own own localized requirements. Certainly, humans could not deal with the complexities of the world unless they partitioned their internal world models into smaller, more manageable, parts.

Localized domain descriptions are modular and easier to understand. They also pave the way to *localized forms of reasoning* – i.e., planning localities individually [10, 11]. It is clear that the cost of expensive planning algorithms can be alleviated by applying them to smaller, localized plans. Localized plans that potentially interact (as determined by domain structure) may then be combined and interactions resolved. GEMPLAN is the only planner I know of that utilizes localized reasoning. It is my hope that the use of domain structure to partition the reasoning space will result in a planner that is able to scale up to large, real-world domains.

1.5 Summary

In summary, the unique properties of multiagent domains suggest a planning architecture (for purposes of multiagent coordination) with the following properties:

- The ability to represent and reason about complex coordination requirements. This will require going beyond traditional planning techniques based on the modal truth criterion (achievement of STRIPS-based conditions) and hierarchical action decomposition. Of course, any multiagent planner should certainly *include* the traditional planning techniques as well (as does GEMPLAN).
- The ability to partition the domain and planning space into smaller localities and combine localized plans effectively.
- The ability to preplan as well as conduct some forms of coordination dynamically.
- The ability to maintain as much correctness as possible, in the face of dynamic, potentially reactive, forms of reasoning.

The GEMPLAN planning architecture described in Section 3 tries to meet these goals.

[2]And, indeed, complex single agent domains.

2 Problems of Past Approaches

In my view, past work in planning has suffered from a myopic view of what planning *is*. Essentially, "planning" has become equated with what traditional planners (e.g., STRIPS [6], NOAH [15], NONLIN [16], SIPE [17], TWEAK [3]) are actually capable of doing, rather than with some more abstract (and more general) view of the task. This has manifested itself in a number of ways.

- The planning process has become equated with the attainment of state-based conditions that are described in terms of STRIPS-like operators. Recent traditional hierarchical planners have also allowed operators to include information about possible action decompositions, and perform planning at increasing levels of detail.

  In my view, planning should be viewed as general purpose *constraint satisfaction*. Here, I am taking the notion of "constraint" to be very broad – essentially, it is any expression the planner is trying to make true. Thus, the "constraint satisfaction" abilities of a planner might include the attainment of state-based conditions, decomposition of actions into subactions, satisfaction of interval-based constraints, assignment of values to variables subject to constraints (CSP-like constraint satisfaction), enforcement of behavioral patterns expressed as regular expressions, reaction to a stimulus, or enforcement of any other constraint the planner knows how to test for and achieve.

- The word "planning" has become synonymous with "pre-planning." As has become increasingly clear, however, much planning must be done dynamically. Thus, reactive and pre-planning techniques must be blended within a complete planning spectrum.

- The notions of planning *search*, *plan modification techniques* (what I call plan *fixes*), and the actual structure of a *plan* have become confused with each other. While there may be connections between the structure of a plan and the search mechanisms and plan modifications used to attain that structure, they should not be equated. One of the most common examples of this is that many planners insist that the ordering of actions in a plan be the same as the order in which those actions are added to the plan by the planner. This, for example, is one of the several explanations for the so-called "Sussman Anomaly."[3] Another similar confusion occurs between the hierarchies within a plan due to nonatomic event decomposition and the hierarchies that exist within the search space.

[3]Despite protests made in a recent paper by Drummond and Currie about the confusion between search and plan [5], they too makes this unstated assumption about action ordering!

A most insidious example is the overwhelmingly popular tendency to confuse the nonlinear (partially ordered) structure of a plan (of course, any multiagent plan *must* be nonlinear) with so-called "nonlinear" search decisions and plan modifications. For example, plan modifications may utilize *least commitment* and leave certain actions, which may ultimately be ordered, unordered for the time being. Alternatively, a plan modification may impose a particular event ordering between two events (perhaps within the context of a larger plan that *is* partially ordered), only to have the planner ultimately backtrack and choose an alternative event ordering.

In GEMPLAN, the notions of search, plan structure, and plan modification strategy (i.e., *fixes*) are quite distinct. The planning process may be viewed as the search of a tree. Each node in the tree is associated with a partially-constructed plan. Each arc represents a plan fix, which results in the new plan at its terminating node. The actual structure of a plan is a partially ordered set of events. Plan fixes may add events anywhere they choose within a plan and, together with search strategies, may adopt least-commitment plan changes or commit-and-backtrack strategies as desired. The search itself (i.e., the order in which plan fixes are applied) is highly general and can be easily tuned by the user.

Just as traditional planning techniques have become fossilized in people's minds, those of us pursuing newer approaches to planning must guard ourselves against the pitfall of claiming one technique for all uses. I believe that the use of reactive planning techniques as exemplified by the work of Kaelbling and Rosenchein on situated automata [9], Chapman and Agre on Pengi [1], and Georgeff and myself on the Procedural Reasoning System (PRS) [7, 8] will find limited application in domains where complex synchronization and coordination must take place. For reasons described in the previous section, coordinative reasoning must often be done in advance, not on the fly. In addition, much of the work in reactive planning (for example, the PRS system built by Georgeff and myself) has no inherent method for understanding plan interactions and resolving them – any coordination must be done by the programmers of these systems when they are designing appropriate reactions. That is, coordination must be explicitly handled by system users rather than by the system itself. Clearly, this is not a realistic requirement for many applications.

Similarly, I believe that coordination by dynamic negotiation and much of the recent work on "communicating rational agents" will not be adequate for domains with complex interactions. While coordination via communication and negotiation certainly has its place (it is how humans generally coordinate among themselves), it is not appropriate for large-scale coordinative problems. This is why workers on the factory floor don't negotiate with each other about how to achieve their tasks and share resources, why large organizations make organizational plans in advance, and why general contractors exist. It is my view that these large-scale coordination problems are the ones that will benefit most from computational aids.

In general, I think that AI researchers have shied away from the complexity that naturally arises from concurrency. They have tended to focus on things like interagent communication and negotiation rather than hard-core coordination algorithms. This is understandable – humans aren't good at dealing with coordination in their day-to-day lives. However, computational concurrency has been an active area of computer science research over the past 20 years and the AI community has a lot to gain by utilizing this work.

3 GEMPLAN Multiagent Planner

In this section I will try to outline the salient features of my research on the GEMPLAN multiagent planner. The current system is implemented in Prolog on a Sun computer and can generate multiagent solutions to several small domain problems. My near-term research goals (described in Section 3.5) include applying the system to a larger, more realistic domain.

3.1 Domain Representation

GEM models a domain as a *set of interrelated events clustered into regions of activity.*[4] Events may be related temporally, causally, or by a simultaneity relation that models required simultaneity.[5] GEM utilizes two kinds of regions: *elements* and *groups.*[6] Each element or group may be associated with explicit constraints on the event behaviors occurring within it. Thus, a domain is viewed as a myriad of interrelated events that are partitioned into regions and subject to regional constraints.

Elements are the most basic type of region. Every event must belong to exactly one element, and all events belonging to the same element must occur sequentially. Thus, elements are loci of sequential activity. Elements (and their constituent events) are then

[4] I use the terms "event," "action," and "event instance" synonymously.

[5] Of course, unordered actions may also potentially occur simultaneously – they are simply not *required* to be simultaneous.

[6] GEM stands for the Group Element Model.

clustered into groups – regions whose boundaries impose barriers on various forms of domain interaction and the influence of constraints. The motivation behind the selection of these particular region types was to mimic the structures found in the world and how people reason about them. For example, at some level, most physical objects in the world are subject to sequentiality constraints. Such regions may be suitably represented as elements. And the fact that humans are capable of dealing with the complexities of the world is evidence that they utilize some form of regional independence – e.g., the kind imposed by groups.

GEM's domain regions may be structured so that they overlap, are disjoint, or form hierarchies – i.e., they may take on *any* structural configuration that the domain describer wishes. The key idea is to partition a domain in such a way that domain properties are *localized*. That is, if a set of events is affected by a specific set of constraints, that set should be represented as a distinct region of activity. A domain's physical structure, its functional decomposition, the extent of causal effect, the processes that may occur — all of these factors may play a role in structuring a domain. Indeed, the regional structure of a domain may reflect many decompositions simultaneously – for instance, a physical decomposition may be overlapped with a functional decomposition.

Syntactically, a GEM domain description is composed of a set of element and group declarations.[7] Each element declaration is associated with a set of event types, describing the kinds of events that may occur within it. A group declaration describes the elements and groups that comprise it. Both element and group declarations are then associated constraints on internal event behavior. These constraints are highly expressive, especially for describing the complex synchronization properties common to multiagent domains.

Again, note that GEM constraints are *localized*, constraining the event behaviors occurring within the element or group with which they are associated. Thus, one of the primary reasons for structuring a domain into elements and groups is to localize the effect of constraints upon event behaviors – i.e., *constraint localization*. The second reason for GEM's locality structures is to impose additional *structural constraints* on events. As already stated, elements impose a sequentiality constraint on their constituent events, and groups limit event interrelationships that cross their boundaries. A more formal definition of GEM's locality properties is provided in [10, 11].

[7]GEM also includes a facility for declaring and instantiating element and group types.

3.2 What is Planning?

As I have stated earlier, I believe that the planning process should be viewed as one of constraint satisfaction – preferably localized according to a natural partitioning of the domain. This is (not surprisingly) precisely the approach taken by GEMPLAN.

Given the structure of a domain and its applicable constraints, GEMPLAN's task is to construct a *plan* (i.e., a set of interrelated events) that is subdivided into regions ("subplans"), each of whose executions satisfies all applicable regional constraints and goal constraints. Thus, GEMPLAN's reasoning process may be viewed as one of constraint satisfaction. Given an initial world plan (possibly empty), the planner repeatedly chooses a constraint, checks to see whether the constraint is satisfied and, if it is not, either backtracks to an earlier decision point in the planning process or continues on, modifying the world plan so that the constraint is satisfied.

This constraint satisfaction process may be seen as the search of a set of trees, one for each region. Each search tree focuses on building a regional plan that satisifies all regional constraints. At each tree node is stored a representation of the currently constructed plan for that region. When a node is reached during planning search, a constraint is checked. To satisfy it, the search space branches for each of the possible ways of repairing or "fixing" the world plan. A *fix* typically involves the addition of new events and event interrelationships; in essence, it massages an ever-growing partial order. The placement of events within this partial order depends on the nature of the constraint and fix – a fix may potentially insert an event or relationship anywhere within the plan it is fixing.

Currently, GEMPLAN is able to satisfy a predefined set of common constraint forms. These include constraints that describe required causal relationships between events, constraints that require sets of events to conform to a behavior expressed as a regular-expression, nonatomic-event expansion, and the achievement of state-based conditions and goals (based on Chapman's *modal truth criterion*).[8] GEMPLAN also includes a facility for *accumulating* constraints on the values of unbound event-parameter variables, similar to the constraint accumulation facility in Wilkins's SIPE system. This allows some forms of least commitment to be utilized.

Of course, the order in which constraints and fixes are applied has an effect on the speed with which a correct plan is found and the actual composition of the plan — indeed, on whether or not any solution is found at all. GEMPLAN tries to provide for as much flexibility as possible in the control of planning search. The order in which constraint checks and fixes are tried (including moves between regional search trees) and decisions about

[8]Thus, the system already exceeds the capabilities of most existing planners.

when and where to backtrack or halt are all guided by user-modifiable *regional search tables*. Whenever backtracking occurs, the node left behind is retained. The search may thus utilize a mixture of depth- and breadth-oriented exploration, depending on the strategy determined by the table. If a user does not supply domain-specific search information, a default depth-first search strategy is used. The constraints and fixes within each region are chosen in the order in which they are supplied within the domain description. Once constraint checking in a specific region is complete, other regions that may be affected are rechecked.

It is interesting to note the differences between this approach to planning and more traditional methods. While most planners can handle only one or two forms of "domain constraints" (attaining state-conditions, nonatomic-event expansion), GEMPLAN can handle any form of constraint, as long as it is furnished a suitable plan-fixing method. Thus, planning "operators" that have been given distinctive status by traditional planners become "constraint satisfaction methods." Second, GEMPLAN's constraint satisfaction search is much more flexible than those found in most planners. Constraints and fixes can be applied in *any* order determined by the regional search tables, which may encode context-sensitive heuristic information.

3.3 The Impact of Locality

The use of locality can have several positive impacts on planning, both from a representational and a reasoning point of view. Representationally, locality provides a means for modularizing the domain description and explicitly stating the scope of effect of domain constraints. Thus, it goes a long way in handling the notorious "frame problem" – determining the extent of effect of events on properties and other events. We have discussed the formal impact of locality on the frame problem elsewhere [10, 11].

In our view, locality will have an even more significant impact on the cost of planning. The partitioning of planning search into localized search spaces can cut planning costs in two ways:

- *Reduction of search space size.*
  Imagine that a domain is represented as a global entity with 10 constraints over its events. At each point in the global search tree, the planner has the potential to choose from 10 constraints to work on – and further, each constraint may be satisfied in multiple ways! This leads to a search space with a branching factor of at least 10.

  Now imagine that the search space is partitioned into two subregions, each with 4 constraints, and a global region with 2 constraints that apply only to the events exported from its subregions. Each subregion tree has a branching factor of approximately 4, and the new global region's tree a branching factor of 2.

This is obviously a substantial reduction in search size. It is also not surprising – "divide and conquer" is a tried and true technique. GEMPLAN provides a way to subdivide the search according to the inherent locality properties of a domain – only relevant constraints are applied within each locality.

- *Reduction of planning algorithm cost.*
  Not only are local search spaces smaller, but the sizes of the plans to which expensive constraint satisfaction algorithms are applied are also smaller. This is because each regional search tree deals only with a smaller, regional plan. In the example described above, even the "global" search space (in the second, partitioned case) deals with a much smaller plan that is composed only of events exported (made visible) by the two subregions. In contrast, traditional hierarchical planners, which may partition the *expansion* of subplans (using hierarchical action decomposition), do not cope with the potential interactions among those subplans in any disciplined manner. As a result, event interactions within a plan must be checked during a costly global interaction analysis. We expect the use of locality to have a substantial impact on the cost of planning.

As far as we know, GEMPLAN is unique in its use of domain structure to partition planning search into localized search spaces. Of course, planning for one region may affect planning within another. The flow of reasoning among regional search spaces must be guided by a domain's potential regional interactions. When plan modifications are made, constraints (and regions) that could possibly be affected (but only those that could be affected) need to be rechecked. In practice, shifts between regional search spaces are enacted by virtue of search table information or the particular nature of a fix. For example, if a group fix adds an event to a subregion, search will be pursued within that subregion.

Given this regional planning architecture, planning complexity is directly related to the amount of regional interaction within a domain — which is as it should be. Planning for tightly coupled regions may involve considerable interaction among their regional search spaces, but planning for domains in which there is little regional interaction will be loosely coupled and could potentially be performed in parallel.

Of course, the ultimate utility of locality is predicated on the assumption that a domain *can* be appropriately subdivided. This will depend on the skill of the domain expert in subdividing a domain description and on the nature of the domain itself. If a domain can be structured so that regional interactions are limited, planning will be cheaper. Tradeoffs must obviously be made between allowing possibly relevant, but costly, interactions and the ultimate tractability of domain reasoning. However, experience thus far with GEM (as well as the fact that humans *are* able to plan in a complex world) is evidence that localizing principles can be effectively used.

3.4 Dynamic Reasoning for Multiagent Domains

In this section I will discuss my plans for extending the GEMPLAN system, which, currently, conducts planning only prior to execution, to a framework that spans preplanning and execution-time (dynamic) reasoning. As I have discussed, the process of planning (deciding upon which actions to execute) is really a spectrum of reasoning techniques that may occur at various times relative to actual execution.

Preplanners tend to produce plans that are provably "correct" but are too rigid and not robust in the face of dynamic changes to the environment. Reactive techniques are more flexible and robust, but tend to be weak in ensuring correctness, especially when it comes to multiagent coordination. In between these extremes lie possibilities for achieving dynamic but well-defined (with respect to plan semantics) plan modification. This includes the use of "synchronizers" – mechanisms that dynamically satisfy constraints at run-time. I expect that they will be especially useful for enforcing priority constraints. Such constraints are quite important in multiagent domains, which tend to resolve resource contention using a priority policy (for example, first-come-first-served). From a realistic point of view, it is often impossible to prioritize resource usage until an actual ordering of resource requests is determined – i.e., during execution.

The full-spectrum planning framework I envision for GEMPLAN will take the following form. Given a domain with its localized constraints and a specific problem to solve, the GEMPLAN will perform as much planning as possible in advance. However, some constraints may be unsatisfiable until run-time. These may include priority constraints, but may also include other constraints due to lack of information – information that can be found only at run-time.

Whenever possible, such constraints should be handled by inserting a "synchronizer" – essentially a run-time demon – into the plan. This synchronizer will perform the appropriate constraint satisfaction at run time, and will hopefully have a well-defined semantics so that reasoning about its possible run-time effects can (if desired) be done during the preplanning phase.

The GEMPLAN executor will then execute the plan, activating "synchronizers" to further control plan execution when necessary. Reactive demons may also be utilized within the execution system. These would perform more typical forms of reactive reasoning, and might be especially useful for dealing with emergency situations, for filling out low-level actions (for example, for navigation), and possibly for plan repair. However, the actual semantic effect of these reactive demons might be less well defined. After several dynamic, reactive repairs, it might ultimately be necessary to fully replan. (As an illustration, a factory floor schedule may be patchable to some extent, but, at some point, may have to be completely rescheduled).

Note that the locality properties of GEMPLAN might be quite useful for dealing with dynamic forms of reasoning – both reactive techniques and full replanning. GEMPLAN stores a link between each event and relation in a plan and the search tree node where it was added. This link can be used for plan explanation as well as plan repair. If a particular event goes awry during execution, GEMPLAN could potentially use the link between the offending event and the node where it was created to guide the planner directly to a point where other strategies might be tried. By knowing which events have already taken place and which events are affected by which constraints (locality), GEMPLAN could determine which events are retractable and which portions of a plan must be repaired.

Similarly, suppose that the set of constraints to be solved suddenly changes. Using the links between events and relations in the old plan and the constraints that caused them to be inserted into that plan, GEMPLAN could determine which events should be removed from the older plan (i.e., those that have not yet occurred and correspond to constraints that were removed). Locality could then be used to distinguish those parts of the plan that will be affected from those that will not. (By the same token, locality can also be used to help determine which parts of a plan may be affected by a reactive demon.) Starting from a newly pared-down initial plan (that saves as much information as possible from the old plan), replanning could then proceed as an instantiation of a new planning problem with a new set of constraints to be solved.

3.5 Goals

My future goals are to further enhance GEMPLAN along the lines described in this report and to apply the system to a large, real-world domain that exercises the system's unique capabilities. Potential applications include a building contractor domain and several NASA applications (for example, scheduling space station experiments so that power consumption needs are coordinated with heat dissipation capabilities, or coordinating the tasks of Mars rovers). Primary areas for further development include:

- Enhancing the constraint satisfaction facility to handle new forms of constraints, including: metric temporal reasoning (work on this began in the summer of 1989), CSP constraints, and priority constraints (i.e., dynamic constraint satisfaction).
- Continuing experimentation with the localized, heuristic search mechanism. The mechanism is currently being enhanced to deal with the full range of locality structures, including regional overlap. Another possible area of exploration is to actually search the various independent local search trees *in parallel.*
- Enhancing the underlying plan structure to include: (1) conditionals and other limited forms of disjunction; (2) synchronizers and reactive demons.
- Building a GEMPLAN executor that integrates the use of synchronizers and reactive demons.
- Applying the system to a domain that heavily utilizes locality and that involves complex forms of coordination.

References

[1] Agre, P. and D. Chapman. "Pengi: An Implementation of a Theory of Activity," in *Proceedings of the Sixth National Conference on Artificial Intelligence – AAAI87*, Seattle, Washington, pp. 268-272 (July 1987).

[2] Allen, J.F. "Towards a General Theory of Action and Time," *Artificial Intelligence*, Volume 23, Number 2, pp. 123-154 (1984).

[3] Chapman, D. "Planning for Conjunctive Goals," Masters Thesis, Technical Report MIT-AI-TR-802, MIT Laboratory for Artificial Intelligence, Cambridge, Massachusetts (1985).

[4] Drummond, M. "Situated Control Rules," in *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, Toronto, Ontario, Canada (May 1989).

[5] Drummond, M. and K. Currie. "Goal Ordering in Partially Ordered Plans," to appear in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence – IJCAI89*, Detroit, Michigan (August 1989).

[6] Fikes, R.E, Hart, P.E., and N.J. Nilsson. "Learning and Executing Generalized Robot Plans," *Artificial Intelligence*, Volume 3, Number 4, pp. 251-288 (1972).

[7] Georgeff, M.P. and A.L. Lansky. "Procedural Knowledge," *Proceedings of the IEEE, Special Issue on Knowledge Representation*, pp. 1383-1398 (October 1986).

[8] Georgeff, M.P. and A.L. Lansky. "Reactive Reasoning and Planning," in *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seattle, Washington (July 1987).

[9] Kaelbling, L.P. "An Architecture for Intelligent Reactive Systems." In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Morgan Kaufmann, Los Altos, California (1987).

[10] Lansky, A.L. "Localized Representation and Planning," The 1989 Stanford Spring Symposium – Workshop on Planning and Search (March 1989).

[11] Lansky, A.L. "Localized Event-Based Reasoning for Multiagent Domains," *Computational Intelligence Journal, Special Issue on Planning*, Volume 4, Number 4 (November 1988). Also appeared as Technical Note 423, Artificial Intelligence Center, SRI International, Menlo Park, California (1988).

[12] Lansky, A.L. "A Representation of Parallel Activity Based on Events, Structure, and Causality," in *Reasoning About Actions and Plans, Proceedings of the 1986 Workshop at Timberline, Oregon*, M. Georgeff and A. Lansky (editors), Morgan Kaufmann Publishers, Los Altos, California, pp. 123-160 (1987). Also appeared as Technical Note 401, Artificial Intelligence Center, SRI International, Menlo Park, California (1986).

[13] Lansky, A.L. "Specification and Analysis of Concurrency," Ph.D. Thesis, Technical Report STAN-CS-83-993, Department of Computer Science, Stanford University, Stanford, California (1983).

[14] Lansky, A.L. and D.S. Fogelsong. "Localized Representation and Planning Methods for Parallel Domains," in *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seattle, Washington, pp. 240-245 (1987).

[15] Sacerdoti, E.D. A Structure for Plans and Behavior, Elsevier North-Holland, Inc., New York, New York (1977).

[16] Tate, A. "Generating Project Networks," in *Proceedings of the Fifth Joint Conference on Artificial Intelligence – IJCAI77*, Boston, Massachusetts (1977).

[17] Wilkins, D.E. "Domain-independent Planning: Representation and Plan Generation," *Artificial Intelligence*, Volume 22, Number 3, pp. 269-301 (1984).