AI Magazine Volume 20 Number 4 (1999) (© AAAI)

CPEF

A Continuous Planning and Execution Framework

Karen L. Myers

■ This article reports on the first phase of the continuous planning and execution framework (CPEF), a system that employs sophisticated plan-generation, -execution, -monitoring, and -repair capabilities to solve complex tasks in unpredictable and dynamic environments. CPEF embraces the philosophy that plans are dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. In particular, plans and activities are updated in response to new information and requirements to ensure that they remain viable and relevant. Users are an integral part of the process, providing input that influences plan generation, repair, and overall system control. CPEF has been applied successfully to generate, execute, and repair complex plans for gaining and maintaining air superiority within a simulated operating environment.

The AI planning community, until recently, has focused its attention almost exclusively on the problem of *generation:* producing a schedule of activities that when performed in some initial state will guarantee achievement of a specified set of goals. With the exception of plan repair, important topics related to the use of plans (robust execution, reactivity, monitoring, evaluation) have received significantly less consideration. In realistic domains, however, plan generation is only a small component of the overall package.

This article reports on the first phase of an effort to develop a system, the continuous planning and execution framework (CPEF), that combines sophisticated plan-generation and plan-use capabilities to solve complex tasks in unpredictable and dynamic environments. CPEF embraces the philosophy that plans are dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. In particular, plans must be updated in response to new information and requirements in a timely fashion to ensure that they remain viable and relevant. Plan execution involves more than blind adherence to previously generated plans. Rather, run-time decisions are made to adapt, initiate, or abandon plans and activities in response to current considerations within the operating environment.

To date, the emphasis for CPEF has been to produce a distributed, multiagent framework in which plan generation and execution are fluidly integrated. The system provides timely adaptation of its activities based on monitoring of critical events within its operating environment. Users are an integral part of the overall process, providing input that will influence the types of plan that are generated, the number of options to consider, failure assessments, plan-repair strategies, and overall control of system behavior.

One unique characteristic of CPEF is that it supports both *direct execution,* in which activities and actions are undertaken by the system itself, and *indirect execution,* in which the system supervises execution of plans by a collection of distributed execution entities. The indirect model of execution is essential for many domains, including work-flow management and many classes of military operations, where software control of plan entities is impossible.

CPEF leverages several sophisticated AI technologies as components. SIPE-2 (Wilkins 1988) provides hierarchical task network (HTN) planning and plan-repair capabilities. The ADVISABLE PLANNER (Myers 1996) supports user provision of advice to guide both plan generation and plan repair within SIPE-2. The ADVISABLE PLANNER thus enables users to direct planning tasks at high levels, letting SIPE-2 manage the underlying details. The procedural reasoning system (PRS) (Georgeff and Ingrand 1989), a knowledge-based reactive control system that integrates goal-oriented and event-driven activity

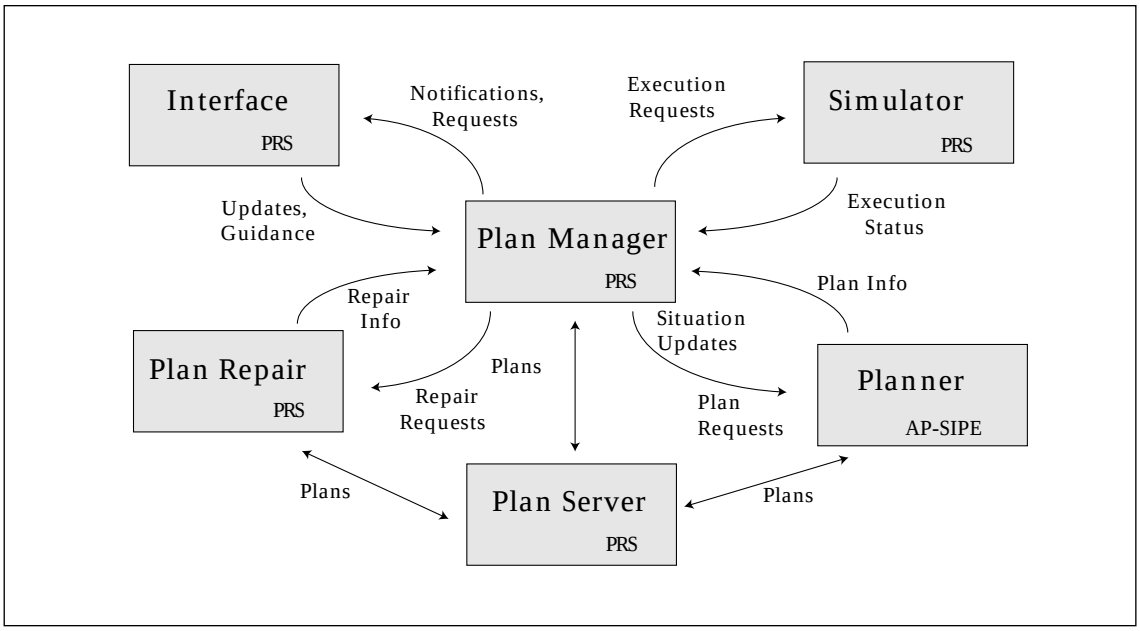


Figure 1. Functional Overview of CPEF.

in a flexible hierarchical framework, is used both as a high-level reactive controller for the overall system and as a way to track execution through generated plans. Finally, CPEF builds on certain capabilities from the multiagent planning architecture (MPA) (Wilkins and Myers 1998) to provide distributed communication and plan-storage services.

CPEF draws on experience gained in building CYPRESS (Wilkins et al. 1995), an integrated planning and execution system that used SIPE-2 as a generative planner and PRS as an executor. Like CYPRESS, CPEF uses a common procedure library (encompassing both plans and operators) encoded in the ACT representation language (Wilkins and Myers 1995). Elements of the library span multiple abstraction levels and can be used for both plan generation and execution, thus supporting smooth transitions between the two capabilities. In particular, plan generation can proceed to arbitrary levels of refinement, with the executor applying additional procedures at run time to refine tasks to executable activities. As with CYPRESS, planning and execution operate asynchronously within CPEF, in a loosely coupled fashion. CPEF components communicate domain knowledge, plans, requests, and situation information, as required, to fulfill their respective responsibilities.

CPEF differs from CYPRESS in several key ways. First, rather than leaving control of the overall system implicit in the activities of the execution module, an explicit *plan manager* oversees operations within the system. Second, CPEF supports a richer set of interactions between the planner and the executor as well as a broader set of plan-adaptation and -repair mechanisms. Third, CYPRESS did not support indirect plan execution, thus limiting its applicability to domains in which planned actions could be executed and monitored by the system itself. Finally, the grounding of CPEF in a powerful multiagent architecture (MPA) enables distributed operations, which were not supported within CYPRESS.

CPEF, although domain-independent technology, is being developed to support a Joint Forces Air Component Commander (JFACC) in the prosecution of realistic air campaigns. This article describes one demonstration that illustrates CPEF's ability to generate, execute, and repair complex air-campaign plans while remaining responsive to changes in guidance and tasking.

CPEF Architecture

Figure 1 provides an overview of the CPEF system. Boxes in the figure represent key functional capabilities, and arrows depict information flow. Attached to each box is the name of one or more technologies—the advisable planner (AP), the procedural reasoning system (PRS), SIPE-2—that implement the associated functions.

CPEF Components

The *plan manager* lies at the heart of the system, directing the overall plan-generation, -monitoring, and -execution processes. The

plan manager is always active, continuously monitoring the world for new tasks and information to which the system should respond.

The *planner* provides core plan-generation and -adaptation capabilities, ranging from fully automated to interactive and *advisable* planning in which users can express recommendations and preferences on the types of plans that are to be produced. Advisable planning is valuable both to support user customizability of generated plans and to enable user-directed exploration of qualitatively different options.

Plan repair oversees adaptations of plans in response to situation changes and execution results. The *simulator* serves as a stand-in for the real-world execution of a plan. The *plan server* provides a repository for storing multiple plans in an organized and principled fashion. Although of limited use within the current system, the plan server will be essential for managing the large numbers of options and subplans required to support long-term continuous planning.

The *interface* supports interactions between the user and CPEF. Users can supply a range of information and requests to the system, including assignment of tasks, situational updates, evaluation assessments, and planning advice. The system informs the user of critical events and activities, soliciting guidance when appropriate to direct problem solving.

Agent-Based Organization

CPEF incorporates portions of MPA (Wilkins and Myers 1998) into its infrastructure. MPA is an agent framework that provides a collection of services and capabilities designed to facilitate the management of complex, distributed planning tasks.

CPEF uses two components of MPA, namely, its communication infrastructure and its plan server. MPA communication consists of a set of message protocols layered on top of KQML (Finin et al. 1992). The protocols define a specialized language for exchanging information and requests related to plans and planning activities. MPA was developed originally to provide general agent services for plan-construction and -evaluation tasks. For MPA to be used as the underlying agent infrastructure for CPEF, the communication protocols had to be extended to support the exchange of requests and results related to plan repair and execution.

Plan Manager

The *plan manager* is responsible for the overall control of system operation. As such, its main responsibilities are to control generation of plans and options for outstanding tasks, oversee execution of plans, provide knowledge management capabilities for plans and plan execution (monitor for key events, perform information-gathering tasks), provide timely response to user requests and unexpected events, and control adaptation of plans in response to plan failures.

The plan manager can execute multiple threads of activity at any given time. This multiprocessing enables, for example, one or more secondary plans to be activated in response to unexpected events while continuing execution of the primary plan (for example, dispatch of a search-and-rescue mission to recover a downed pilot during execution of the main air-campaign plan).

Direct versus Indirect Execution

Previous work on reactive execution of plans has focused on models in which an executor *directly* performs the activities specified in a plan. For example, the software controller for a mobile robot would initiate actual execution of actions by the robot (turning, increasing speed, or stopping) in the physical world.

This direct model of execution is inappropriate for the JFACC domain because the actions in JFACC plans must be performed by the pilots, soldiers, marines, and support staff who are involved with the campaign. Rather, the JFACC operating environment requires an *indirect* model of execution in which plan activities are performed by human agents in the real world rather than a software controller. Within this indirect model, the role of an *executor* is to *track* the execution of the plan rather than to carry out the actions. Tracking involves monitoring progress through the execution of the plan based on information (possibly incomplete) about the outcomes of individual actions within the plan.

Although a seemingly subtle distinction, the difference between direct and indirect execution greatly impacts the design of an executor. With indirect execution, an executor might not have immediate access to information about the success or failure of prescribed actions or might not even be able to determine whether these actions ever took place. Even when information is available, there can be a significant time lag between performance of an action and receipt of information about its status. Similarly, there can be substantial delays and cost in redirecting activities of the agents that are performing the actions in the plan.

The plan manager uses a *flow model* for tracking plan execution. This approach involves waiting for reports on the outcome (success, failure, or unknown) of individual actions, in

accordance with the temporal ordering relationships of actions in the plan. For example, if action *A*1 precedes action *A*2, the flow model dictates that the outcome of *A*1 must be determined and appropriate responses taken before the outcome of *A*2 can be considered. This tracking mechanism was implemented as a variant of the standard hierarchical task-execution mechanisms within PRS. In particular, specialized methods were defined for action achievement and condition testing that implement tracking rather than execution semantics. In the future, our group (see Acknowledgments) intends to explore more opportunistic tracking models that enable response to action outcomes in arbitrary orders.

Monitors

Within the AI community, models for detecting and recovering from plan-execution failures have generally been limited to precondition failures and action failures.

The creation and deployment of *monitors* is a critical part of CPEF. A monitor is an event-response rule for which detection of the specified event leads to execution of the designated response.

We have begun development of a taxonomy of monitor classes that will enable appropriate measured responses to critical detected events. The different classes derive from variations in both the type of event to be detected and the nature of the response. We believe that this classification will enable simpler and more modular specifications of monitors for both automated and interactive approaches.

The main categories of concern to date have been *failure, knowledge,* and *assumption monitors:* Failure monitors encode appropriate responses to failures that could occur during execution of a plan. Knowledge monitors test for the availability of information about a world-state condition that is needed for decision making. Assumption monitors test for situation changes that violate assumptions upon which a given plan relies. Assumption monitors are particularly valuable in that they enable early detection of potential problems with a plan rather than waiting for problems to surface during plan execution.

CPEF supports user definition of a wide range of monitors. In addition, it provides automated generation of assumption monitors based on the content of a plan. The algorithm for extracting assumption monitors involves a traversal of the HTN plan-derivation structures, collecting from each node those operator applicability conditions that are *dynamic* (for example, troop locations and threat levels but not geographic conditions) and are not included in the effects of some preceding node in the plan (that is, they must be satisfied in the initial world). The effects-based filtering eliminates large numbers of conditions that should not be monitored because they are to be established by actions within the plan. Prespecified domain models identify a response to be performed when the applicability conditions are violated. Currently, there are three categories of responses: (1) alerts for the user, (2) plan repairs, and (3) invocation of standard operating procedures. In addition, the domain models indicate conditions for which assumption monitors are definable, thus filtering conditions whose violation is not significant. For example, weather conditions can fluctuate over time; their status can be disregarded until entry into a critical time window preceding key actions.

Failures and Repairs

Within CPEF, the plan manager determines when to initiate modifications to a plan. In contrast to many systems, individual failures do not necessarily lead to plan repair. Rather, the plan manager supports a variety of models for interpreting failures and responding.

Generalized Failure Models

Within the AI community, models for detecting and recovering from plan-execution failures have generally been limited to *precondition failures* and *action failures.* A precondition failure arises when associated preconditions for an action are not satisfied at the time the action is to be executed. An action failure results when the execution of an action does not attain its intended effects. These two types of failure, although important, cover only a small portion of the space of possible failures. The initial CPEF system takes a first step toward a more general framework through its accommodation of the following additional failure types.

Unattributable Failures Failures are called unattributable if no individual action has failed or no assumption is violated, but some assessment (human or automated) has deemed the current plan inadequate. For example, a commander might declare that a planned breach of the enemy's air-defense system has failed, despite the success of each constituent mission. Such a situation can arise either because the planning operators do not model the real world with sufficient fidelity or simply because the commander has a conservative nature (for example, he/she requires a high guarantee of neutralization before he/she is willing to fly subsequent missions through a sector).

Aggregate Failures In many situations, a single failure need not be cause for alarm.

Indeed, good human planners often build redundancies into their plans to improve robustness. As a concrete illustration, air-campaign plans often include extra missions above and beyond what is required to satisfy the objectives at hand to improve the likelihood of success. An aggregate failure is defined by a set of actions whose collective failure constitutes a significant event.

Detection of unattributable and aggregate failures requires information beyond what is stored in plan-dependency structures. Within CPEF currently, unattributable failures are identified by human assessors, and a simple domain-specific theory of aggregate failures has been defined for air-campaign plans.

Plan Repairs

The JFACC application domain (like many others) requires the use of *conservative* repairs (Nebel and Koehler 1995) that minimize changes to the original plan. Plans should evolve gradually, with small changes in the world or current goals resulting in proportionally small changes to the plan. Minimization of changes is important to ensure the continuity of the plan and because of the potentially high costs of redirecting execution entities. To date, the focus in CPEF has been on conservative repair based on analysis of plan-dependency structures (Kambhampati and Hendler 1992; Wilkins 1985). In particular, CPEF relies on the core methods defined previously within SIPE-2, along with several extensions that support more flexible forms of plan repair.

Generally speaking, plan repair based on dependency structure analysis involves identifying a set of *root nodes* that are the source of failures. Each such root has an associated wedge of lower-level tasks, which are removed from the plan. New subplans are then generated for each root node, if possible; otherwise, the process repeats for the parent of the node, terminating when the generation process succeeds. To support the repair of unattributable failures, CPEF allows users to identify arbitrary root nodes whose wedges are to be replaced.

A second extension to the methods in SIPE-2 enables replanning for *task generator* nodes. A *task generator node* differs from standard task nodes in that it spawns a set of instances of a task template rather than a single task instance. The set of instances is determined by a special *creation condition:* An instance of the task template is created for each set of bindings that satisfies the creation condition. Generator nodes provide a powerful representational capability that is critical for planning in many realistic domains. The operator knowledge base for the JFACC domain relies extensively on generator nodes. For example, it contains an operator named Protect-all-threatened-COGS that can be applied to reduce threat levels to blue centers of gravity.[1] The operator contains (among other subgoals) the following goal generator:

```
ACTION: GENERATE-GOALS
GOAL-GENERATOR: (defend-cog
   threatened-place when1 rating1)
CREATION-CONDITION: (blue-cog
   threatened-place)
```

To support plan repair, generated tasks whose creation conditions are violated are identified and removed from a plan. Furthermore, situation changes that result in the satisfaction of additional instances of the creation conditions lead to insertion of corresponding generated tasks into the plan.

The plan-repair capabilities within CPEF represent a start toward more flexible plan-adaptation mechanisms. More work is required to produce the general plan-repair framework envisioned for the final CPEF system. Methods grounded in the analysis of dependency structures produce a plan that is proven *correct* with respect to the underlying domain model; here, correctness means that simulated execution of the plan will result in a world state where the original goals are satisfied. Even though this notion of correctness is somewhat weak (because unexpected events generally will occur, and the domain knowledge itself might be faulty), guarantees of correctness can be computationally expensive to secure. For this reason, a continuous planning system should provide a spectrum of plan-repair mechanisms ranging from the correct but costly minimal-perturbation, dependency structure methods to transformational approaches that use domain-specific repair rules (in the spirit of Ambite and Knoblock [1997]), possibly trading correctness for efficiency.

Simulation Environment

The JFACC application domain precludes evaluation of CPEF in an actual operational setting. Furthermore, no appropriate simulation environment exists in which to conduct experiments. For these reasons, a PRS-based simulation environment called simulated flexible execution (SIMFLEX) was developed to enable testing, evaluation, and demonstration of the continuous planning and execution capabilities of CPEF.

SIMFLEX provides an "epsilon-fidelity" simulation capability that neither requires elaborate domain-specific action models nor tracks state information in the simulated world. The

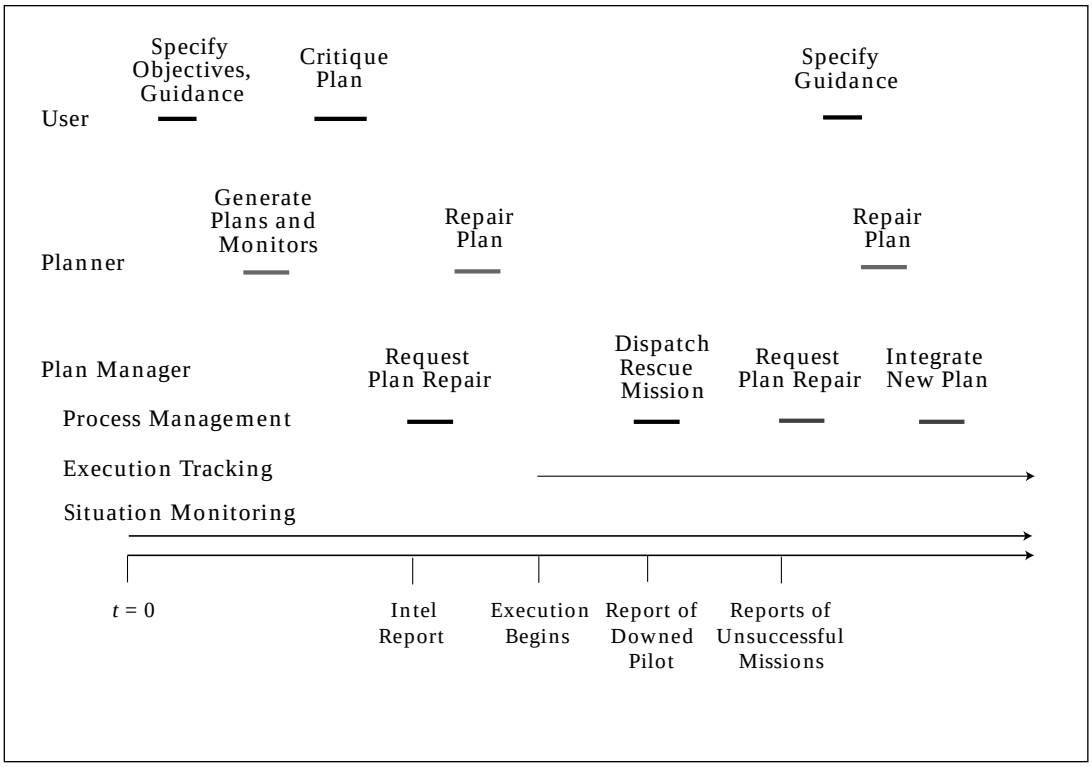


Figure 2. Demonstration Timeline.

only required input to SIMFLEX is a description of the set of actions and conditions for a given domain. For each possible action, SIMFLEX generates an *act* (Wilkins and Myers 1995) for simulating the execution of the action. Similarly, SIMFLEX generates an act for simulating the testing of each condition in the domain. These acts are parameterized, thus enabling runtime–modifiable outcomes for the actions and conditions in the plan. To simulate a given plan, SIMFLEX traverses through its nodes, invoking the generated acts using the standard task-refinement methods within PRS. By building on the infrastructure of PRS in this manner, development of the simulation environment required relatively little effort.

User-specifiable distributions determine both the rates at which actions or condition tests succeed, fail, or yield no information about their execution, as well as the duration of actions. Users can also specify overriding success and duration rates for individual actions, conditions, and tasks. As such, SIMFLEX provides a flexible, tailorable environment in which to conduct evaluation of continuous planning capabilities.

CPEF Application

CPEF has been applied to an air-campaign planning domain, with emphasis on achieving air superiority within a designated region. The plans in this domain are derived through hierarchical refinement of objectives for defensive and offensive air superiority, terminating at the level of the missions to be flown (Lee 1998). The final plans, which require less than a minute to generate, contain several thousand nodes and can span over 30 refinement levels.[2]

Figure 2 provides a high-level overview of a partial run of the system. The timeline along the bottom shows exogenous events that lie outside the system's control. Above this timeline, activities are organized along three functional roles: (1) the user, (2) the planner, and (3) the plan manager. The system is designed so that the user plays an active role in both plan development and execution. The term *planner* is used generically to refer to a number of planning-related activities: plan generation, plan analysis, and plan repair. The plan manager is active at all times, providing three main threads of activity in parallel: (1) situation monitoring, (2) execution tracking, and (3) process management.

Activity begins in response to the user specifying air objectives for a given campaign, along with advice that reflects the commander's guidance for one or more courses of action to be developed. The planner generates plans to satisfy these objectives and advice, relative to its current knowledge of the operating environ-

ment. The completed plans are reviewed by the user, who can recommend changes to satisfy any outstanding concerns through the specification of additional or modified advice; the planner then produces updated plans that incorporate the user's feedback. (Alternatively, the user could request the generation of a plan with markedly different characteristics through the specification of different advice.) As planning proceeds, the plan manager monitors the environment for events that are relevant to the developing plan. Receipt of an intelligence update that invalidates parts of the plan will result in the planner being tasked to repair the affected portions.

Monitoring of the environment continues as execution of the selected plan commences. At some point during the execution, notification is received that a pilot has been downed; the system responds by instigating an appropriate activity (for example, a search-and-rescue mission). In addition to monitoring for such critical events, the plan manager tracks progress through the execution of the plan to determine whether modifications are needed in response to the status of the execution. As an illustration, one generated plan contains missions to neutralize a set of surface-to-air missile sites as a way of enabling access to a critical air sector. Receipt of reports indicating that more than a designated threshold of missions failed (an aggregate failure) triggers plan repair to address the failure. The user can provide advice to influence the kinds of modifications that are made to the plan during the repair process. For example, the user can recommend a strategy that involves the establishment of a second neutralization mission or a more radical approach that eliminates the need for access to the air sector.

Conclusions

CPEF, although a work in progress, already provides many of the foundational capabilities required for continuous planning and execution in highly dynamic and complex worlds. These capabilities include an agent-based architecture, rich monitoring and repair strategies, flexible integration of plan generation and execution, and highly adaptive problem-solving capabilities. Much more is required, however, to produce a truly continuous planning and execution system.

One key area for future research is open-ended planning. CPEF currently relies on traditional planning methods that create end-to-end solutions. Continuous operation requires the ability to produce open-ended plans that grow and evolve in response to the dynamics of the environment. Our group is exploring methods for generating plans whose abstraction depth and temporal extent are grounded in the knowledge and constraints of the current situation. Incremental planning techniques will be required to enable the growth and evolution of open-ended plans in response to situation changes.

Acknowledgments

This research was supported by the Defense Advanced Research Projects Agency, contract F30602-97-C-0067, under the supervision of Air Force Research Lab-Rome. David Blei, Tom Lee, and David Wilkins contributed toward the development of CPEF.

Notes

1. A *center of gravity* corresponds to a location or capability that has been designated as critical either to defend (blue) or attack (red).

2. Extensive documentation for the demonstration system, including sample plans, is available at www.ai.sri.com/~cpef/jfacc.html.

References

Ambite, J. L., and Knoblock, C. A. 1997. Planning by Rewriting: Efficiently Generating High-Quality Plans. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 706–713. Menlo Park, Calif.: American Association for Artificial Intelligence.

Finin, T.; Weber, J.; Wiederhold, G.; Genesereth, M.; Fritzson, R.; McKay, D.; and McGuire, J. 1992. Specification of the KQML Agent-Communication Language. Technical Report EIT TR92-04, Enterprise Integration Technologies, Palo Alto, California.

Georgeff, M. P., and Ingrand, F. F. 1989. Decision Making in an Embedded Reasoning System. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 972–978. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Incorporated.

Kambhampati, S., and Hendler, J. 1992. A Validation-Structure–Based Theory of Plan Modification and Reuse. *Artificial Intelligence* 55(2): 192–258.

Lee, T. J. 1998. The Air-Campaign Planning Knowledge Base. Technical Report, Advanced Automation Technology Center, SRI International, Menlo Park, California.

Myers, K. L. 1996. Strategic Advice for Hierarchical Planners. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, eds. L. C. Aiello, J. Doyle, and S. Shapiro, 112–123. San Francisco, Calif.: Morgan Kaufmann.

Nebel, B., and Koehler, J. 1995. Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis. *Artificial Intelligence* 76(1–2): 427–454.

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. San Francisco, Calif.: Morgan Kaufmann.

Wilkins, D. E. 1985. Recovering from Execution Errors in SIPE. *Computational Intelligence* 1(1): 33–45.

Wilkins, D. E., and Myers, K. L. 1998. A Multiagent Planning Architecture. In *Proceedings of the Fourth International Conference on AI Planning Systems*, 154–162. Menlo Park, Calif.: AAAI Press.

Wilkins, D. E., and Myers, K. L. 1995. A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation* 5(6): 731–761.

Wilkins, D. E.; Myers, K. L.; Lowrance, J. D.; and Wesley, L. P. 1995. Planning and Reacting in Uncertain and Dynamic Environments. *Journal of Experimental and Theoretical AI* 7(1): 197–227.

Karen Myers has been a member of the research staff at SRI International since completing her Ph.D. in computer science at Stanford University in 1991. Her research interests focus primarily on planning, reactive control, agent-based systems, and design rationale. She was a key designer and developer of the multiagent planning architecture for distributed planning and the CYPRESS system. She has pioneered work on allowing users to control automated planners through the provision of advice, as exemplified in the ADVISABLE PLANNER system. Her work on reactive control has included the development of controllers for mobile robots, real-time tracking, and crisis action management. Her e-mail address is myers@ ai.sri.com.

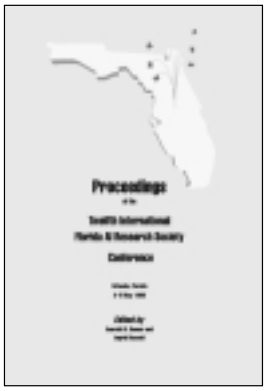
New Proceedings from AAAI Press

Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology

Edited by Thomas Lengauer, Reinhard Schneider, Peer Bork, Douglas Brutlag, Janice Glasgow, Hans-Werner Mewes, and Ralf Zimmer

The ISMB conference series provides a general forum for disseminating the latest developments in bioinformatics. ISMB is a multidisciplinary conference that brings together scientists from computer science, mathematics, molecular biology, and statistics. Its scope includes the development and application of advanced computational methods for biological problems. Relevant computational techniques include machine learning, pattern recognition, knowledge representation, databases, combinatorics, stochastic modeling, string and graph algorithms, linguistic methods, robotics, constraint satisfaction, and parallel computation. Biological areas of interest include biomolecular structure, genomics, biomolecular sequence analysis, evolution, and phylogenetics, biomolecular interactions, metabolic pathways, regulatory networks, developmental control, and molecular biology generally. Emphasis is placed on the validation of methods using real data sets, on practical applications in the biological sciences, and on development of novel computational techniques.

ISBN 1-57735-083-9 324 pp., illus., references, index
$45.00 softcover



Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference

Edited by Amruth Kumar and Ingrid Russell

The Florida AI Research Society Conference was founded in 1987 to promote and advance AI research within the state of Florida, fostering interaction between researchers at colleges, universities, and industry. Since 1990, Florida AI Research Society conferences have been broadened to include participants and papers from across North America and the world. This year's proceedings covers a wide range of topics, including applications, computer vision, evolutionary computation, intelligent agents, knowledge-based systems, learning and AI, logic and AI, logic programming, natural language processing, planning, AI applied to spacecraft autonomy, evolutionary computation, intelligent tutoring systems, knowledge management, neural network applications, parallel and distributed reasoning, reasoning about function, spatiotemporal reasoning, uncertain reasoning, and verification, validation, and knowledge-base refinement.

ISBN 1-57735-080-4 560 pp., illus., index
$50.00 softcover

Published by the AAAI Press
445 Burgess Drive
Menlo Park, California 94025

To order, call 650-328-3123. http://www.aaai.org/Press/