Control of Search in AI Planning

In the field of artificial intelligence, "planning" is defined as designing the behavior of some entity that acts, either an individual, a group, or an organization. The output is some kind of blueprint for behavior, which we call a plan. There are a wide variety of planning problems, differentiated by the types of their inputs and outputs. Typically, planning problems get more and more difficult as more flexible inputs are allowed and fewer constraints on the output are required. As this flexibility increases, the space of possibilities needing to be explored by the planning algorithm grows extremely quickly (usually exponentially).
The goal of this workshop was to provide a specific focus on controlling this search. This problem is the essence of the planning problem, and it arises regardless of which specific planning methodology is used (nonlinear planning, deductive planning, hierarchical planning, etc.) - in all of these controlling an exponentially growing search space is a central problem. In all planning formalizations, it is critical that some sort of knowledge (heuristic or otherwise) is used to make reasonable decisions at any of the many choice points which arise in planning. Such choice points can concern:

- ordering of subgoals
- selection of operators/control structures
- resolution of conflicts/threats by different techniques
- choosing between differing commitment strategies
- selecting/choosing the right control regime

In all of these cases, picking the right control knowledge can result in an algorithm that is able to identify and prune many dead-end branches of the search space before the algorithm explores them, ideally while preserving the soundness and completeness of the planner. However, designing search control approaches is difficult and it is often impossible to ensure various qualitative and quantitative properties of the "controlled algorithms".

The workshop brought together over 30 participants from Europe, America, Asia, and Australia representing a large part of the planning community. The programme consisted of invited survey talks, short presentations of participants, working group sessions, and a general discussion about the empirical evaluation of planning systems. We had invited talks on *search control in deductive planning*, *structural categorization of planning domains and problems*, *dynamic planning*, and *search problems and techniques from operations research*. The working groups addressed in detail the questions of different search spaces, domain based features and domain analysis, and search control in dynamic real time domains.

This report contains the abstracts of presentation and invited talks as well as a summary of the working group sessions.

Jim Hendler and *Jana Koehler*, November 1996

Deductive vs. "Classical" Planning or How to Waste Research Efforts

by *Wolfgang Bibel*, Technical University Darmstadt, Germany (also affiliated with University of British Columbia).

So-called "classical" planning has largely ignored its natural ties with automated deduction (AD). Even a logic-oriented AI book as the one by Russell and Norvig (1994, p. 342) sets planning apart of AD. This talk is a reminder of these close ties and points out the potential benefits of such a joint approach.
As a logical basis we take Bibel's linear connection method (1986), which is a practically important fragment (and in fact anticipator) of linear logic (1987). Therein we demonstrate with three well-known concepts in classical planning (partially ordered plans, causal links and protection) that they are just different names of well-known concepts in AD (spanning matings, isolated connection reduction, irreversibility of reductions). These concrete examples together with general theoretical knowledge show that searching for plans is exactly the same as searching for proofs, even on a low level. It is therefore waste of research efforts if the two communities develop their techniques independently.
Finally, we give a short overview of the substantial body of research (documented in numerous publications) which has been carried out mainly in Germany on deductive planning with resource sensitive logics such as the linear connection method.

Search Control in Deductive Planning

by *Susanne Biundo*, German Research Center for Artificial Intelligence, Germany.

Deductive planning systems rely on an expressive logical representation formalism, a proper formal semantics, and a calculus the rules of which are used to implement the planner. Plans are generated by constructive proof of so-called plan specification formulae. In the simplest case, specification formulae describe the initial state and the goal state and demand for the existence of a plan that transforms the one into the other. Starting from a plan specification, a proof (tree) is developed by applying logical inference rules in a backward-chaining manner, and by instantiating the plan variable accordingly. We have introduced the modal temporal planning logic TPL, as an example, and have demonstrated how deductive planning works in this context. TPL is an expressive formalism that allows for the distinction of rigid and flexible symbols and provides a programming language for plans, including control structures like conditionals and loops. The search problems and solutions we have addressed are concerned with the guidance of the theorem proving = planning process, the selection of appropriate basic actions, and the selection of subgoals. Since a carefully designed domain model is an essential prerequisite for acting safely and efficiently in this context, the planning environment we have introduced provides deductive support in setting up provably consistent domain models.

Constraint-Based Planning Architectures

by *Amedeo Cesta*, IP-CNR, Rome, Italy.

The general context for our research is the design of integrated planning and scheduling architectures for constraint-based activity management. A number of aspects useful to improve the applicability of such architectures have been addressed. One aspect is connected with the involvement of users in actively exploiting such systems. Relevant problems are: the definition of a domain representation language powerful enough to represent the physical constraints of a domain, and endowed with clear semantics as a basis for automated verification tools; the investigation of various aspects of the interaction with the user like the "cognitive intelligibility" of the plan representation and the planning process. A second aspect consists in the study of specialized classes of constraint propagation techniques: interesting results on the efficiency and flexibility of manipulating quantitative temporal networks have been obtained through the synthesis of dynamic algorithms for constraint posting and retraction; also the problem of mixed resource and time constraints representation has been studied and some techniques proposed for the synthesis of implicit temporal constraints from the analysis of resource representation. A further aspect consists in the integration of the incremental constructive way of building a solution with local search techniques: in particular a taboo search algorithm has been proposed that take advantage of the given temporal representation to solve planning problems requiring multiple resources.
Joint work with Angelo Oddi and Cristiano Stella.

Propositional STRIPS Planning as the Semantic and Computational Foundations for AI Planning

by *Tom Dean*, Brown University, USA.

I believe that the propositional STRIPS planning problem (or a suitable generalization thereof that allows for slightly more representational freedom) constitutes a very important object of inquiry for AI. I am not advocating STRIPS as the sole object of study for AI planning; however, I think that it deserves considerable attention given that it captures the one particularly important aspect of AI planning problems: the dynamics can be represented using *poly log* n space even though all n states are reachable from the initial state. AI has from the earliest been interested in how a small set of rules can be used to represent very complex interactions. We know a little about the asymptotic worst-case complexity of STRIPS and its variants, but there is still a great deal that we don't know. In particular, we know little about approximations and even less about expected performance measures. It is too bad that STRIPS is not currently as compelling as say TSP. However, I would claim that it is every bit as interesting from a mathematical standpoint as TSP and I believe that compelling applications will emerge as practitioners become more knowledgeable about planning technology and this technology matures.

Dynamic Planning: A cause or solution to problems in planning search control?

by *Edmund H. Durfee*, University of Michigan, USA.

Plans need to be executed by real systems, operating in real worlds. The world won't wait while the system decides what to do, and the system might have only limited computational and sensory resources to make its decisions. The goal of planning, therefore, is to generate a plan specification that when executed works fast enough within the limited resources to avoid undesirable states and accomplish desired goals. But such plans might not exist! To deal with this case, the planner needs to eliminate some demands - which means that the system might be unprepared for some situations. Dynamic planning is the solution to this problem: when "out of its depth" the system can rely on the planner to plan a way out of the difficulty. Of course, this means that the system has to be watching out for situations that it is unprepared for (it must "expect the unexpected"), and be able to replan fast enough to preempt catastrophic consequences of being outside of the range of states for which the system is prepared. This means that dynamic planning is the cause of the need for controlling search, since planning cannot take too long. Fortunately, the planner might use the same models of the domain for planning its replanning actions as for planning the actions in the world, and therefore can determine the timing needs of replanning. This ability allows it to plan for dynamic planning that is fast enough to assure some level of desired performance.

The Search Control, the Holy Grail of Planning?

by *Brian Drabble*, University of Edinburgh, United Kingdom.

Intelligent planning systems are now being applied to ever more challenging real world problems and tasks. However, as the complexity of the problem domain increases so does the need to embed more planning knowledge in the system. Without knowledge to guide the planning system in its choice of "what to do next" and in pruning invalid branches in the search space the size of the problem becomes intractable. While some people may view this as "cheating" it is an essential element in any practical planning solution.

The O-Plan project has pioneered a number of search control techniques which aim to prune the search space or provide guidance on which issue to address next. These techniques include:

1. *Typed Preconditions*: which encodes the user's intention on the ways in which a precondition should be satisfied.

2. *Branch1/BranchN*: which provides a measure of the impact on the search space at the next level (Branch1) and at the most primitive level (BranchN).

3. *Issue Dependencies*: which provides an explicit triggering language to allow issue to be released for processing at the most opportune time.

As planning problems increase in size and complexity the need for further and more expressive representation techniques will increase. The challenge to AI planning researchers is to develop these expressive representation while at the same time maintaining the search contol power needed to solve challenging real world problems.

Memory-limited Search and a Workbench for Heuristic Search

by *Jürgen Eckerle*, Albert Ludwigs University Freiburg, Germany.

State space search has a wide range of applications, often only restricted by space and time constraints. Especially best-first search is a useful search paradigm in the area of heuristic search. Classical best-first search like the well-known A*, however, has a space requirement which grows exponentially with search depth since each generated node is stored. For that reason several best-first search algorithms have been developed in the recent years which are admissible under space limitations.
One method for efficient usage of memory is provided by our algorithm DBIDA* (dynamic balanced IDA*) which on the one hand makes detecting duplicates more worthwhile than in former approaches and on the other hand reduces the cost for updating the data structure. The additional overhead cost per node expansion (compared with IDA*) amortized over a sequence of expansions is quite small. This is achieved by combining breadth-first and depth-first search with strategies for balancing the explicitly stored subpart of the search tree.
Results explaining the tradeoff between space and time complexity are useful. For a general model of best-first state space search including e.g. IDA*, ITS, SMA*, DBIDA* etc. we can prove a lower bound for the time complexity of space-limited search. This lower bound is sharp in case of duplicate free search trees. We prove that the product of space and time (measured in the number of node expansions) is of order $\Omega(n^2)$ in the worst-case where n is the number of states in the problem graph.
Furthermore, we present our program environment called HSF a general workbench for solving state space problems based on heuristic search. HSF allows both a fair comparison of different search algorithms under common conditions as well as rapid prototyping of new algorithms. The environment has been successfully used for a practical course in heuristic search in which two-person teams implemented and visualized one selected puzzle.

Agent-based HTN Planning

by *Kutluhan Erol*, Intelligent Automation Inc., USA.

This work is focused on studying and evaluating a planning architecture that can incorporate expertise and data from a variety of sources. It will support planning and execution simultaneously, recover from failures and exploit opportunities, and also support on-line job requests from multiple users. Our proposed architecture is based on autonomous agents, and it employs hierarchical task network (HTN) planning techniques for handling the interactions and resolving the conflicts among the agents. HTN techniques have provided a significant improvement over the traditional state-based techniques by focusing on jobs (referred to as tasks in the HTN terminology) and on the interactions among them.

It will combine the advantages of domain-specific planning by capturing specialized expertise in the agents (which will be the task experts) and the advantages of domain-independent planning by providing an architecture that can be easily implemented, incrementally modified, and adapted to the changing conditions. Wherever the machine intelligence does not suffice, human operators can be integrated to the architecture as agents, interacting through computer terminals.

Reactive Action Plans for Task Execution

by *R. James Firby*, University of Chicago, USA.

I am interested in building intelligent agents to act in everyday environments. To cope with the complexity of such domains, the agent typically needs to plan at an abstract level, neglecting many of the details involved in actually interacting with the world. However, this leaves the agent with the problem of executing plans that do not contain actions that can be directly executed in the world. My Reactive Action Plan (RAP) system for task execution is a model for how that process might take place. Planned tasks are expanded at run time using a hierarchical library of task methods that include mechanisms for synchronizing actions with continuous control processes, interpreting sensor and action results in context, monitoring task progress, and retrying actions that do not succeed in achieving their goals. In addition, the RAP language is specifically designed so that RAP methods can be treated as abstract planning operators. The RAP system has been used for task execution in a variety of soft real-time systems including on our robot at the University of Chicago. For the robot, the RAP system has been tightly integrated with a modular, reconfigurable vision and control system.

Planning in Uncertain Worlds

by *Michael Georgeff*, Australian Institute for AI (AIAI), Australia.

In today's technological world, computers are increasingly finding application in complex process control, business management, provision of customer service, medical treatment, telecommunications, and so on. These systems often need to carry out quite complex tasks, over some period of time, in a world subject to change and uncertainty. The very scale of the application often prevents complete specification, and even if this could be accomplished, the business and social drivers usually demand changes to the specifications even before the first implementation is rolled out. In the world in which we live, chaos, uncertainty, and change are the norm, not the exception. Despite this, most designers of complex realtime systems continue to apply software technologies and methodologies that were constructed for static, certain, and definable worlds. This invariably leads to huge time and cost overruns, dramatic system failures, and relentless pressure on IT providers to accommodate change and ill-specified functionality.

Our research focus is on these types of application: dynamic, uncertain, and complex. We aim to investigate and design software architectures suited to such applications and to develop methodologies that allow the construction of systems that can survive uncertainty and change.

To this end, we have been conducting research and development on agent-oriented systems (also commonly known as software agents) for over ten years. Agent-oriented technology is a fundamentally new paradigm for building computer systems that we believe will come to dominate the area of distributed realtime systems.
An agent-oriented system consists of a collection of autonomous software agents (possibly a very large number), each of which responds to events generated by other agents and by the environment in which the agents are situated. Each agent continuously receives perceptual input and, based on its internal state, responds to the environment by taking certain actions that, in turn, affect the environment.
Our theoretical research has examined the logical foundations of agent architectures (such as the mental attitudes of beliefs, desires, goals, intentions, plans, commitments, and obligations) and the processes that operate on these mental attitudes (such as deliberation, means-end reasoning, and reconsideration). Using a possible worlds framework, we have formalized various mental attitudes and the processes that operate on them.
We are also building tools to assist the design, development, and deployment of software agents. We have developed agent-oriented systems such as dMARS, a C++ based software development environment for building and testing software agents. The system development effort has been supported by research into areas that bridge the gap between theory and application, such as abstract architectures for building software agents and agent-oriented languages.
We have also worked closely with end-users to produce a range of large-scale agent-oriented applications in areas such as air traffic management, business process management, and air-combat modelling. Many of these applications are fully operational systems, and provide important feedback for the further development of dMARS and raised important questions for foundational research.

Accelerating Partial-Order Planners: Some Techniques for Effective Search Control and Pruning

by *Alfonso Gerevini*, University of Brescia, Italy.

We are concerned with improving the performance of "well-founded" domain-independent planners – planners that permit proofs of soundness, completeness, or other desirable theoretical properties. Recently, we have proposed some domain-independent techniques for accelerating partial-order planners. The first two techniques are aimed at improving search control while keeping overhead costs low. One is based on a simple adjustment to the default A* heuristic used by UCPOP to select plans for refinement. The other is based on preferring "zero commitment" (forced) plan refinements whenever possible, and using LIFO prioritization otherwise.
A more radical technique is the use of operator parameter domains to prune search. These domains are initially computed from the definitions of the operators and the initial and goal conditions, using a polynomial-time algorithm that propagates sets of constants through the operator graph, starting in the initial conditions. During planning, parameter domains can be used to prune nonviable operator instances and to remove spurious clobbering threats. While we applied this technique in the context of partial-order planning, it could also be applied to other planning frameworks.

In experiments based on modifications of UCPOP, our improved plan and goal selection strategies gave speedups by factors ranging from 5 to more than 1000 for a variety of problems that are nontrivial for the unmodified version. Crucially, the hardest problems gave the greatest improvements. The pruning technique based on parameter domains often gave speedups by one order of magnitude or more for difficult problems, both with the default UCPOP search strategy and with our improved strategy. The Lisp code for our techniques is available from the authors or as an on-line appendice of a paper recently published in JAIR vol 5, 1996.
This is joint work with Len Schubert (University of Rochester, USA).

Planning for Autonomous Mobile Robots

by *Malik Ghallab*, LAAS-CNRS Toulouse, France.

Planning is reasoning on actions, of different kinds and at different levels. The autonomous robot domain, as explored by the Robotics and AI group at LAAS-CNRS, offers a rich paradigm for investigating and experimenting with planning problems in dynamic worlds. Particular actions such as those involving the management of sensors and actuators have been modeled with their specific characteristics, e.g. uncertain, imprecise and non-deterministic effects, geometrical or kinematic constraints. Specific planners have been developed and integrated within robots; they take into account complex planning operators, e.g. closed-loop motion control with sensor feedback, relying on domain specific as well as on more general planning techniques. A domain independent planner, called IxTeT, relies on these specific planners and the PRS-based reactive level. IxTeT handles explicitly time and events expected in a dynamic world. A flexible architecture, now on board on 4 different robots developed by the group, eases the integration and offers a high level of robustness, through standardization and formal specification and code generation of modules. The functional level of this architecture is composed of specific planners as well as of low level control modules; the supervision level in charge of the real-time decision making is the PRS-based execution monitoring and reacting system; it controls the IxTeT planning level.
Several open problems that arise for planning in such a domain are actively being worked on by the group. Among these, the relationship between the planners and the reactive procedures (today hand-coded) need to be better understood and made more flexible. A decision theoretic planner (today restricted to perception planning and decoupled from the task planner) would handle the uncertainty in the domain knowledge; it arises the issue of consistent heterogeneous planning.

On Executing Discrete POMDP Plans

by *Joachim Hertzberg*, GMD Sankt Augustin, Germany.

Assume that a plan is to be executed in the presence of uncertainty in the sensor information and in the action control and that the execution is required to pass through certain given situations of the environment. We discuss invariant properties that plan execution under these assumptions should obey. For discrete Partially Observable Markov Decision Problems (POMDPs) as the formal framework underlying plan execution, our approach provides a plan execution monitoring algorithm, which is derived

from a path following monitor of a mobile robot. This algorithm is then shown to fulfill the required properties of plan execution.

The Design and Implementation of a Derivational Replay Framework with Learning from Case Failure

by *Laurie H. Ihrig* and *Subbarao Kambhampati*, Arizona State University, USA.

Case-Based Planning (CBP) provides a way of scaling up domain-independent planning to solve complex problems in realistic domains. It replaces the detailed and lengthy search for a solution to a large problem with the retrieval and adaptation of previous planning experiences. In general, CBP has been demonstrated to improve performance over generative (from-scratch) planning. However, the performance improvements it provides are dependent on adequate judgements as to problem similarity. In particular, although CBP may substantially reduce planning effort overall, it is subject to a mis-retrieval problem. The success of CBP depends on these retrieval errors being relatively rare.

DerSNLP (Derivational Systematic NonLinear Planner) is a case-based planner which performs eager derivation replay. DerSNLP extends current CBP methodology by treating case failure as a learning opportunity. It incorporates explanation-based learning techniques that allow it to explain and learn from the retrieval failures it encounters. These techniques are used to refine judgements about problem similarity in response to feedback when a wrong decision has been made. The same failure analysis is also used in building the case library, through the addition of repairing cases. An empirical evaluation of this approach demonstrates the advantage of learning from case failure.

Improving the Learning Efficiencies of Realtime Search

by *Toru Ishida*, Kyoto University, Japan.

The capability of learning is one of the salient features of realtime search algorithms such as LRTA*. The major impediment is, however, the instability of the solution quality during convergence:

1. they try to find all optimal solutions even after obtaining fairly good solutions, and
2. they tend to move towards unexplored areas thus failing to balance exploration and exploitation.

We propose and analyze two new realtime search algorithms to stabilize the convergence process. Epsilon-search (weighted realtime search) allows suboptimal solutions with epsilon error to reduce the total amount of learning performed. Delta-search (realtime search with upper bounds) utilizes the upper bounds of estimated costs, which become available after the problem is solved once. Guided by the upper bounds, delta-search can better control the tradeoff between exploration and exploitation.

Efficient Planning by Graph Rewriting

by *Craig A. Knoblock*, University of Southern California, USA.

Planning involves the generation of a network of actions that achieves a desired goal given an initial state of the world. There has been significant progress in the analysis of planning algorithms, particularly in partial-order and in hierarchical task network (HTN) planning. In this abstract we propose a more general framework in which planning is seen as a graph rewriting process. This approach subsumes previous work and offers new opportunities for efficient planning.

A (partial) plan is a labelled graph whose nodes are actions and whose edges express constraints (ordering, causal links, etc). In planning by graph rewriting we allow the substitution of an arbitrary partial plan by another partial plan. This subsumes the main transformations present in partial-order planners (adding a new node or linking to a previous one for goal establishment, adding ordering edges for threat resolution), and in HTN planners (substituting a non-primitive action by a partial plan). There are several planning domains that can benefit from expressive plan transformations (such as query access planning and machine-shop process planning). We expect that hill-climbing from a (possibly suboptimal but easily constructed) initial plan using such transformations will be efficient for many domains.

Joint work with Jose Luis Ambite.

Finding Optimal Solutions to Rubik's Cube using Memory-Based Heuristics

by *Richard E. Korf*, University of California at Los Angeles, USA.

We present the first optimal solutions to random instances of Rubik's Cube, a problem with about 4×10^{19} states. The branching factor of the problem space is about 13.35, and the median optimal solution length is 18 moves. The heuristic search algorithm used is Iterative-Deepening-A* (IDA*). Traditional heuristic functions, such as 3-dimensional manhattan distance, are not sufficient to optimally solve this problem on current machines. The heuristic we employ is a large lookup table that contains the number of moves required to solve just the corner cubies, from each possible position and orientation, which is a lower bound on the number of moves required to solve the entire puzzle. This table contains about 88 million entries, occupies 42 megabytes of memory, and is easily constructed from a breadth-first search of the subspace containing just the corner cubies. A complete set of 10 random problem instances were solved optimally. If N is the total number of states in a problem, M is the number of heuristic values that can be stored in memory, and T is the number nodes generated by IDA*, then we can show that T is approximately equal to N/M. This means that the running time of the algorithm decreases linearly with the amount of available memory. In our experiments with Rubik's Cube, for example, $N = 10^{19}$, $M = 10^{8}$, and $T = 10^{11}$. As computer memories get larger and cheaper, this algorithm will become increasingly cost-effective.

Combinatorial optimization and frequency assignment: an overview of algorithmic approaches

by *Jan Karel Lenstra*, Eindhoven University of Technology/CWI, The Netherlands.

Frequency assignment problems occur when a network of radio links has to be established. Each link has to be assigned an operating frequency from a set of available frequencies. The assignment has to satisfy certain interference limiting restrictions defined on pairs of links. Moreover, the number of frequencies used is to be minimized. These problems have been investigated by a consortium consisting of research groups from Delft, Eindhoven, London, Maastricht, Norwich, and Toulouse. The participants developed optimization algorithms based on branch-and-cut and constraint satisfaction, as well as approximation techniques such as simulated annealing, taboo search, variable-depth search, genetic algorithms, and potential reduction methods. These algorithms were tested and compared on a set of real-life instances.

Towards intelligent agents in production planning and adaptive scheduling

by *Hans-Jakob Lüthi*, ETH Zürich, Switzerland.

The paper describes the use of an agent framework within the context of a planning and control system for an automated work cell. Because of the dynamic complexity of the planning task, this system must be able to monitor itself, and to react as independently as possible to changing external circumstances. In order to achieve this, the planning task is distributed among a number of virtual intelligent agents. Each one of these agents supports a clearly defined task, has its own problem-solving strategies, and can request assistance from other agents. Simulation plays an important part in this system. It is used by the agents to monitor their own solution strategies, and to assess the future consequences of their decisions.

Controlling Means-Ends Analysis Search Using Regression-Match Graphs

by *Drew McDermott*, Yale University, USA.

The classical planning problem is to find a sequence of actions to achieve a goal from an initial situation, given definitions of the actions in a STRIPS-style representation. There are several search spaces one can use to solve such problems. One of the simplest is the set of *plan prefixes,* or sequences of actions that the solver hopes to extend to a complete solution. This space has not been attractive because of the lack of good heuristic estimators of the distance to the nearest solution from a given plan prefix. One promising idea is to build a *regression-match graph* in order to analyze the structure of a problem. This is a graph whose nodes are ground literals, linked to actions that might achieve them. The preconditions of each action are matched to the current situation to obtain a set of difference literals that must be achieved by further actions, and so forth. The *estimated effort* associated with a literal is 0 if the literal is currently true; otherwise it is the sum of the estimated efforts for the difference literals of its most promising action. For some problem classes, using the regression-match graph transforms an exponential search into a polynomial one.

For other problems, the regression-match graph does not work so well, and the reason often is that the graph gives an effort of 0 to literals that are currently true, even if they are sure to be deleted soon. It looks, however, as if it ought to be possible to predict using the graph which deletions are inevitable. In some simple cases things do work out that way. It is a current research goal to see if the idea can be made to work efficiently in a broader set of cases.

Island Planning and Refinement in Proof Planning

by *Erica Melis*, University Saarbrücken, Germany.

Proof planning is an alternative to classical theorem proving. Proof planning is classical planning with no goal interaction. It faces search problems because of long solutions and possibly infinite branching. Furthermore, plans that are comprehensible to the user are required in particular in mixed-initiative proof planning.
For planning proofs by induction a specific meta-level control has been introduced. This is not enough, however, and does not work for all kinds of proofs. Inspired by proof planning examples, we use a planning framework that can employ *multiple planning strategies*, such as forward/backward state-space and HTN-planning. In order to tackle the search-space-problem and the structured-plan-problem, new planning strategies are introduced and integrated into the general planning framework: *island planning* and *island refinement*, as well as *subproblem refinement*.
The planning with multiple strategies has the additional choice point of selecting an appropriate planning strategy. Some exemplary control knowledge for choosing strategies is developed.
We think that planning for other realistic problems, even in a static and deterministic environment and with complete information, can make use of our ideas.

AI Planning: Theory Versus Practice

by *Dana S. Nau*, University of Maryland, USA.

AI planning systems are finally reaching the point where they can provide useful solutions to practical problems. Using Hierarchical Task Network (HTN) planning techniques, we are developing a planning system that plays contract bridge better than the best available commercial program, and a concurrent-engineering system for integrated design and process planning of microwave transmit/receive modules.
We are using an atypical control strategy for HTN planning: rather than doing goal-directed partial-order planning, we instead do total-order planning using a depth-first left-to-right search. We have found this approach preferable for two reasons:

- Several of the domain assumptions that might lead one to prefer goal-directed search and partial-order planning are inapplicable in our planning domains (and we suspect they are inapplicable in many other practical planning domains).
- With our approach, the planner knows the input state of each step of the plan.

This makes it possible to do complex numerical and probabilistic computations, and to interact with external information sources – both of which are necessary for planning in our domains (and many other practical domains).

Solving Hard Qualitative Temporal Reasoning Problems by Using Tractable Subclasses of Allen's Interval Algebra

by *Bernhard Nebel*, Albert Ludwigs University Freiburg, Germany.

While the worst-case computational properties of Allen's calculus for qualitative temporal reasoning have been analyzed quite extensively, the determination of the empirical efficiency of algorithms for solving the consistency problem in this calculus has received only little research attention. In this talk, we demonstrate that using the ORD-Horn class in Ladkin and Reinefeld's backtracking algorithm leads to performance improvements when deciding consistency of hard instances in Allen's calculus. For this purpose, we identify phase transition regions of the reasoning problem, and compare the improvements of ORD-Horn with other heuristic methods when applied to instances in the phase transition region. Further, we give evidence that combining search methods orthogonally can dramatically improve the performance of the backtracking algorithm. While these result probably do not have immediate consequences for planning, they raise the questions whether similar phenomena are observable in planning problems. In particular, it would be interesting to find a phase transition region for planning and to try to apply tractability results for planning in order to speed up planning algorithms.

SATPLAN — Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search

by *Bart Selman*, AT&T Laboratories, USA.

Planning is a notoriously hard combinatorial search problem. In many interesting domains, current planning algorithms fail to scale up gracefully. By combining a general, stochastic search algorithm and appropriate problem encodings based on propositional logic, we are able to solve hard planning problems many times faster than the best current planning systems. Although stochastic methods have been shown to be very effective on a wide range of scheduling problems, this is the first demonstration of its power on truly challenging classical planning instances. This work also provides a new perspective on representational issues in planning.
Joint work with Henry Kautz.

Deductive Planning in Real World Applications

by *Werner Stephan*, German Research Center for Artificial Intelligence, Germany.

In Deductive Planning, planning problems are considered as deduction problems and are solved by using some theorem prover to construct a correctness proof and the plan itself hand in hand.
During the last years, we have developed the formal and technical basis for applying this approach to real world situations, thereby exploiting similiarities with formal software construction, among others.
Formalisms which we have used in this context are (special variants of) Dynamic Logic and Temporal Logic. Recent work includes:

- the development of basic techniques for axiomatizing planning domains,
- the formalization of strategies for plan generation (special refinement strategies, for example), and
- the development of mechanisms for structuring (the axiomatization of) planning domains.

For the actual process of proof generation (= plan generation) we follow the paradigm of Tactical Theorem Proving. This allows for structuring proofs in way such that (macro-) steps correspond to meaningful manipulations in the underlying scenario.
Current work is concerned with the formalization and implementation of (generic) search strategies used in conventional (non-deductive) planners. Bridging this gap would allow for a combination of efficient search techniques with the advantages of Deductive Planning.

Reactive Planning: Failure, Sensing and Planning Actions

by *Paolo Traverso*, IRST, Italy.

I am interested in systems which are able to execute and to construct plans of actions which may fail, actions that acquire information from the real world, and actions that generate and execute plans of actions. These planning systems must take into account the fact that, as it happens in real systems, actions may fail, and must provide the ability of reasoning about failure handling in acting, sensing and planning. A long term goal of my research is to provide a formal account of systems which are able to plan to act, plan to sense and plan to plan, and therefore, to integrate action, perception and reasoning. Some recent developments of this research has lead to the construction of a "Model Based Planner", i.e. a planner where goals, systems and plans are described as state transition graphs. This allows for the application of efficient techniques of planning search.
Joint work with Alessandro Cimatti, Enrico Giunchiglia, Fausto Giunchiglia, and Luca Spalazzi.

Control of AI Search in Planning Through Reuse of Experience

by *Manuela Veloso*, Carnegie Mellon University, USA.

For several years now, we have developed several methods to acquire control knowledge for AI planning through learning from experience. The Prodigy planner has a well-defined decision cycle which is open to be controlled by invocation of any guidance. In particular, we have pursued the use of derivational replay of one or multiple past planning episodes to reduce the search in complex planning problems. Recently, we applied the strategy of derivational replay to a realistic route planning problem using the real map of Pittsburgh. Planning cases are indexed and retrieved based on geometric and symbolic features, e.g. time of day, relative location, user preferences. Derivational replay has been also combined with conditional planning. Planning effort is shared among different conditional branches; extra steps are added if needed, as well as unnecessary steps are identified and deleted. The similarity between conditional

branches makes the derivational replay produce considerable savings in search in planning. We recently have started research on planning and real execution. We combined Prodigy with a robotic platform Xavier. The integrated architecture, Rogue, controls the planning for asynchronous user requests and monitors their execution with the goal of learning to improve planning from execution experience. Finally, we are also pursuing the development of layered learning and planning in a multiagent system, such as robotic soccer. We have used a simulator and a real robotic framework to study issues of collaborative and adversarial planning in robotic soccer.

Multiagent Planning Architecture

by *David E. Wilkins*, SRI International, USA.

The objective of our Multiagent Planning Architecture project is to develop a new architecture for large, sophisticated planning problems that require the coordinated efforts of diverse, geographically distributed human and computer experts. We have defined a Multiagent Planning Architecture (MPA) that will facilitate incorporation of new tools and capabilities as they become available, and exploration of new ways of reconfiguring, reimplementing, and adding new capabilities to the planning system. In particular, this project will explore novel strategies for integrating planning and scheduling.

MPA is a layered, agent-based architecture that groups planning agents (PAs) and meta-PAs into Planning Cells. We will build upon the diverse range of planning capabilities already developed under the DARPA-Rome Laboratory Planning Initiative (ARPI), drawing from the JTF-ATD architecture while extending its capabilities for planning.

MPA will provide wrappers and agent libraries (in both C and Lisp) to facilitate the construction of agents from legacy systems. PRS-CL, a reactive execution system originally developed for NASA, provides the technology for our most sophisticated agent wrappers. PAs will be defined for several classes of ARPI technology, and will communicate using the common plan representation and the agent communication languages already being developed by other ARPI projects.

MPA provides specifications of the messages to be sent between agents. Thus, all PAs of a common class would outwardly be the same, except for differences in the range of generic services provided; inwardly they would employ application-specific data structures and techniques.

In addition, we will "program" a community of agents in MPA that will apply new technologies to air campaign planning. In particular, we will decompose an ARPI planning technology and an ARPI scheduling technology, so that each technology becomes a set of PAs. This decomposition will allow previously distinct ARPI technologies to be more tightly and flexibly integrated, and allow other technologies to replace some of the PAs in a modular fashion. We will use meta-PAs to define different control strategies among the PAs.

WORKING GROUPS

Working Group 1: Different Search Spaces

This working group included Wolfgang Bibel,Tom Dean, Drew McDermott, Alfonso Gerevini, Jim Hendler, Laurie Ihrig, Craig Knoblock, Richard Korf, Erica Melis, Bart Selman, Werner Stephan, Manuela Veloso.
The challenge for this working group was to identify a unifying framework in which we can compare the various planning approachs and their corresponding search spaces. Some of the existing approaches and systems include:

- Forward and backward search
- Total-order planners (e.g., Strips, Prodigy)
- Partial-order planners (e.g., UCPOP, Tweak)
- Deductive planners
- Graphplan & SATplan
- Abstraction planning
- Case-based planning

Initial Framework
We started with the obvious approach and attempted to characterize all of these approaches in terms of the space they search. Thus, we attempted to describe each approach in terms of the nodes and transitions between the nodes that define the space.

- Forward and Backward Search – This is often what people refer to as state space search. The nodes are a description of the situation or world state and the transitions are the actions that transform one state into another.

- Total-order Planners – This includes planners such as Prodigy and Strips, which use a means-ends analysis search. For these planners the nodes consist of a partial initial plan along with a goal stack or goal set and the transitions are either an operator application or a subgoal.

- Partial-order Planners – This includes planners such as UCPOP and Tweak, which are planners that search through a space of partially-ordered plans. The simple characterization of this space is to view the nodes as partial plans and the transitions as commitments (either ordering constraints or new steps) to the plan. The more accurate characterization is that the nodes describe possible completions and the transitions are restrictions on the completions. The difference is that there may be additional data structures and refinements that are not captured by the simple characterization.

- Deductive Planners – After some discussion, we concluded that the search space of the deductive planners is equivalent to the partial-order planners. The nodes are possible completions and the transitions are restrictions on the completions.

- Graphplan & SATplan – These are planners that on the face of it are searching a very different space. However, as noted previously by Rao Kambhampati, one way to view these systems is that they provide a approach to representing a disjunction of partial plans. Thus, the nodes in the space are disjunctive partial plans and the transition are restrictions on the completions of these plans.

One limitation of this framework is that it does not capture the preprocessing that many systems perform before actually searching one of these spaces. For example, in the case of Graphplan, a key to its efficient search is to propagate mutual exclusion constraints in order to reduce the size of the search space. Similarly, abstraction or problem reformulation systems first search in the space of problem reformulations before they attempt to actually search the ground space.

Key Research Issues
The working group spent some time discussing what we thought were some of the important research topics that future work should focus on:

- How to control search?
  - avoid redundancy
  - generate evaluation functions
- How to represent problems?
  - problem space
  - space that is searched
- How to produce high quality plans?
- How to put planning into scheduling?
- How to combine search techniques? e.g., HTN and abstraction planning
- How to compare different approaches?
- Is there a mathematical generalization of the STRIPS representation to cover more practical problems?

Interesting Questions
We discussed a great many topics and not all of them can be discribed in a coherent story, so I have compiled a few of what I think are the more interesting points that were raised at some point in the dicussion.

- What are the tradeoffs in different languages?
  - speed
  - expressability
  - ease of use
- Is it more efficient to search the plan space or the state space?
  - may be easier to express search heuristics in one formalism
  - state space planners may be forced to commit early
- How can STRIPS-style operators be used to represent numerical computations?

- How can we express knowledge in terms of a set of rules that describes a complex space?
  One of the useful features of the STRIPS-formalism is that it can be used to represent an exponential space in a compact form.
- Case-based planning can be viewed a a form of change in the search space.
  There is an adaptation space instead of search space. There is also a space-time tradeoff between memory and time.
- How many cases do you need for case-based reasoning to cover a space?

Working Group 2: Search Control in Dynamic Real Time Domains

The working group discussion has been organized along the following main steps:

1. a characterization of the domain, the environment and the system specifications where dynamic real time search control is needed,
2. a characterization of the main issues that systems planning in such domains should address and of the main kind of functions, properties and solutions they should provide,
3. a list of open research problems, possibly with a practical research guideline and plan for the short or long-term future.

Domain – environment – system characteristics
Some of the problems that planners in dynamic domains have to address are the following: timing requirements, uncertain-imprecise-incomplete information about the world, change of goals to be achieved and of data/information/facts available from the external environment. Moreover, planning systems have usually to satisfy a complex set of domain dependent requirements, e.g. the ability to recover from failures and unexpected situations, robustness and reliability, safety and liveness requirements.

Issues, tasks, and key factors
In a planning system for real time dynamic domains there are two main kinds of different planning/search activities:

1. Planning *for* dynamic domains: this corresponds to the (off-line) search for constructing a reliable/robust/safe plan which can drive the system during execution.
2. Planning *in* dynamic domains: this corresponds to the search activities which have to be performed during plan execution in order to achieve goals and meet requirements.

An analysis of the difference between planning *for* and *in* dynamic domains cannot be conducted independently of the fact that systems working in real time domains have to deal not only with search, but also with execution. The main problem is not only a problem of search control, it is also a problem execution control. These two kinds of main activities are strictly related and cannot be dealt with in a completely separate way. According to this perspective, the input parameters to the construction of a system executing and planning for/in dynamic domains are the following:

1. plan time vs. action time
2. plan time vs. perception time
3. plan time vs. change rate
4. variability of 1.-3.
5. costs/rewards

What is actually available to the planner is a set of low level behaviors (programs, automata,..) which control low level actions and sensor readings. These behaviors have to be matched by the planner w.r.t. the goals to be achieved and the current context the system is working in. There are different planning approaches one can undertake: (1) synthesize universal automata (off-line), (2) synthesize specific automata, (3) synthesize abstract plan based on the set of available behaviours.
In (2) and (3), at any decision point, the planner must decide whether to act or to (re)plan. More in detail, at each decision point, the planner has to decide whether to continue the current plan, sense to improve the state model, or retrieve or synthesize another plan/automaton.
One of the key issues is how to decide for one of the different behaviors above. In this decision, some key factors are the quality of the plan w.r.t. robustness, optimality, action cost, the time cost vs. quality, and the plan context, i.e. its effects on the time/quality curve. In certain cases, depending on time constraints, a search mechanism which is not optimal but fast enough, might be better than optimal planners. Depending on the application, the trade off between optimality (quality) of plans and time cost is a crucial factor.

Open research problems

Get Empirical!
A main short term goal is to provide a set of benchmark problems which can be used to evaluate dynamic planners. Some of those proposed by the working group are the following: Autoline, AARIA, Truckworld, ACM flight simulator, robot simulator, Forest Fire Fighting Simulator, Civilization, Air campaign simulator, Robocup in Football. A longer term goal is to find a set of abstract and artificial domains. The benchmarks should take into account both search and execution. One of the main issues here is to find out a set of evaluation criteria and metrics for the evaluation of planners.

Search + Execution control.
A key issue is to study the interaction between plans and reactive procedures and the problem of the control of the interactive loop between the planner and the executor.

Heterogeneous planning and representation.
Planning in a dynamic domain needs the integration of different, heterogeneous planning mechanisms and representations.

Pixels to predicates: an intermediate level.
A key problem is to find a proper intermediate representation level from the data structures manipulated by the low level sensor and actuator controllers and the higher level data structures needed for controlling planning search.

Robustness, stability, perpetual.
Several potential application areas of planners for dynamic domains require to deal with the problem of guaranteeing a robust and stable behaviour.

Building user confidence.
The industrial take up of planning in dynamic domains is strongly dependent on the capability to provide a degree of confidence to end-users that the planner will behave safely and satisfactorily under unespected events and situations.

Designing planning that anticipate dynamic replanning.

Working Group 3: Domain based features and domain analysis

The members of the group included Brian Drabble, Amedeo Cesta, Yannis Dimopoulos, Susanne Biundo, Dana Nau, and Jana Koehler. Our group produced two things: a set of observations, and a list of open problems. In both of these, we only included things that the entire group agreed on—we even discussed the wording of our conclusions to make sure everyone agreed.

Observations
The knowledge acquisition problem is different for different planning representations. This hasn't been generally recognized, because of the limitations of STRIPS operators—STRIPS operators lead people to believe that it is easy to represent planning domains, and encourage people to make idealized assumptions that lead to shallow representations.
If it could be done correctly, it would be useful to have a basic planning ontology, whose basic building blocks can be used to build any representation of plans. In other words, this would be an ontolingua for planning analogous to KIF, the Knowledge Interchange Format. However, developing this will be very difficult, because for planning, we need to select a good representation for computation. One question is how to represent control knowledge for a domain—control knowledge depends not only on the domain but also on the planning algorithm.
In developing such an ontolingua, it will be difficult to avoid being

1. too problem-dependent to be general,
2. general but too vague to be useful.

One possible approach might be to have representation formalisms that can be refined depending on what representational details are needed—a library of different representation techniques, and ways to combine them together in a way that insures a well-founded semantics—along with information about how this will affect the representational power and the complexity of planning, and the completeness fo the algorithm. Here are several other points to keep in mind in carrying out such an effort:

1. It would be a very good idea to look at standardization efforts in other fields.
2. Three kinds of planning knowledge: problem specific, domain specific, and domain independent. In order to make planning systems easy to use and maintain, we need to be clear about which is which. Part of why SNLP became popular

was because it was so easy to understand: it has a small easy-to-describe domain-independent strategy, and an easy-to-describe operator syntax. Other planners that were more complicated to describe have not been as widely accepted, partly because they were difficult to describe.

3. It is desirable to have a way to encode integrity constraints into the domain representation. This can be quite useful for debugging purposes.

4. In most planners, is relatively easy to modify the planner to generate not only the plan but also the tactical information it used to develop the plan. This information, if generated, can be helpful for replanning.

Open Problems

1. Is there a way to automatically generate pruning strategies by analyzing the planning domain?

2. How to judge adequacy of a representation?

3. How to measure and describe domain complexity?

4. Does there exist a representation that is more expressive than STRIPS, and easier to use (for finding adequate representations) than O-Plan and temporal logic?

For problem 4, we discussed the trade-off between representational power and computational efficiency of the planning algorithms. We agreed that different representation formalisms can be ranked in a partial order w.r.t. their representational power, with STRIPS being rather weakly expressive and O-PLAN and temporal logic being very expressive. The computational efficiency of STRIPS planning algorithms was rated rather low, while O-PLAN is rather highly efficient since a lot of control information can be incoded in the representation.
Another trade-off exists between representational power and the ease of finding an *adequate* representation of a domain. Here, none of the three representation techniques can be considered as making it easy to come up with a really adequate representation of a domain.