Integrating Cognitive Systems First intermediate report: Project outline

Kai von Luck Bernd Owsnicki-Klewe

FH Hamburg
FB Elektrotechnik und Informatik
Berliner Tor 3
D-20099 Hamburg

{luck,owsnicki}@informatik.fh-hamburg.de

October 16, 1995

Integrating
Cognitive
Systems

Abstract

Purpose of this paper is the outline of the ICS project. At the end of the first period of thinking about integrating cognitive systems into a complex system for a specific task first results are given. First of all the selected task (i.e. real world planning) and a scenario is sketched guiding the next steps if the ICS project. Open problems are mentioned and a way of adressing some of these problems is given. The body of resources available for our project is listed and shortly discussed at the end of this paper.

Contents

1 Project outline

The ICS project (short for **I**ntegrating **C**ognitive **Systems**) aims at researching theoretical and practical methods to combine software and hardware components into one system that can solve a wider range of problems than standard systems.

Special emphasis has been put on the cognitive motivation of system components, a term that will be detailled in section 2. Components like these have shown their usefulness in many applications, but still suffer from lack of integration, both towards each other and with standard software products, like databases and user interfaces.

1.1 Objectives

The objectives of the ICS project are twofold:

1. Investigation of mechanisms to integrate different concepts and, if possible, software modules into one system, espcially mechanisms from the area of Artificial Intelligence. The resulting system should be able to solve tasks of varying difficulty and prepare for industrial cooperations.
2. Construction of a modular, orthogonal collection of software modules to support experiments in the areas of Software Engineering and Artificial Intelligence.

1.2 Report outline

This report is structured as follows: We will first clarify the notion of *Cognitive Systems* and will discuss possible selections from this field for further investigation. We will then sketch the notion of *planning* that will be central to the ICS project. After these preliminaries we will describe the project scenario, i.e. the collections of tasks building the framework for the project. This is followed by a detailed description of the research and development subtasks that will be treated during the project. A final collection of available resources concludes this report.

2 Cognitive Systems

Besides the standard repertoire of Computer Science problem solutions, like Software Engineering, databases, user-interfaces, etc. several approaches exist that are motivated by human problem solving methods and perception abilities. Among others, the classical field of Artificial Intelligence explores the utilization of these techniques for solving tasks inaccessible to standard software techniques.

In [Pylyshyn, 1984] the idea of looking at human cognitive capabilities as a system of modules was first elaborated. A comprehensive list of these cognitive capabilities might look like this:

- Understanding natural (typed or spoken) language
- Producing natural (typed or sythesized) language
- Understanding visual and tactile information
- Manipulating objects
- Memory
- Deduction and problem solving
- Adaption (e.g. learning)

2.1 A technical view

Instead of concentrating on the foundations from Cognitive Science, we adopt a more technical view. That means the ICS project will adopt elaborate methods from the repertoire of actual Computer Science research and development, rather than striving for totally new developments.

We will use the term *Cognitive Systems* in referring to methods from the following areas:

- Symbolic methods, like knowledge based systems, Expert Systems, learning systems, plan generation and execution systems.
- Sensor data interpretation, especially image, language and speech recognition.
- Motoric and effectoric control, as in robotics.
- Approximative and sub-symbolic methods, e.g. Neural Networks, Fuzzy Logic, Genetic Algorithms.

2.2 Integration

The realization of the different methods indicated above show a distinct variety, leading to certain problems when attempting to integrate them into a complete problem-solving system. Even within the framework of symbolic reasoning alone, open problems remain.[1]

2.3 A selection of topics

The ICS project will select several of these problems and methods and integrate it into a complete scenario serving as an problem environment to be researched in depth (cf. section 4).

While it's out of the question to try to construct a system embodying *all* components, the ICS project defines a wider framework allowing to investigate different integration issues. The problems involved in such wide a scope of research and development requires a careful construction of the scenario in order to allow for an incremental definition of increasingly more complex subtasks.

The basic scenario will be built around a autonomous agent (i.e. robot), equipped with the appropriate hardware and software, that is able to solve well-defined task in a simulated or real environment.

This scenario poses several problems to be tackled:

Acting: What is the repertoire of actions this agent can employ?

Planning: How can these actions combined into a reasonable sequence to solve a given problem?

Execution: How can this sequence of actions be executed?

Monitoring: How can this execution be monitored to cope with unexpected events?

Sensing and effecting: What is the interface of the agent to the real world? How does the evluation of sensor data affect the planning, execution and monitoring of the agent's behavior?

Architecture: How do different modules interact in such a system?

In the next section we will briefly sketch the current state of the art in planning.

[1] E.g. the authors have dealt with the integration problem of various concepts into terminological knowledge representation systems, as in [Owsnicki-Klewe *et al.*, 1993; von Luck, 1989; Heinsohn and Owsnicki-Klewe, 1988].

3 Planning

The notion of planning needs careful scrutinity to avoid confusion with colloquial use and to lead to a clear distinction in terms of tasks a planning system may be confronted with.

In [Tate *et al.*, 1990] a taxonomy of AI planning techniques is given. We will contrast here some important streams covering the field, starting from the classical perspective and finally will discuss more recent approaches.

3.1 Planning as reasoning about actions and plans

Georgeff [Georgeff, 1990] sets the general framework of *reasoning about actions and plans* and continues to present the central notions of this field.

> "...the world is in one of a potentially infinite number of *states* or *situations*.
>
> ...
>
> The world can change its state only by the occurrence of an *event* or *action*.
>
> ...
>
> An *action* is a special kind of event — namely, one that is *performed* by some agent, usually in some intentional way."

From this setting several problems arise that require in-depth studies:

World representation: What constitutes the properties of the environment relevant for the planning task? How shall these properties be represented and maintained?

Action representation: What constitutes the applicability and the effects of performing a certain action?

Intention: How can the agent's intention be represented?

Plan structure: What constitutes a plan?

Plan generation: How can the agent's intention be transformed into such a plan?

Complexity: How expensive in terms of time and memory is this process of plan generation?

Correctness: Is the resulting plan correct, i.e. does it satisfy the agent's intention?

Optimiality: Is the resulting plan optimal wrt. given criteria?

Executability: Can the resulting plan be executed in reality?

The general formal framework employed uses symbolic representations, like logical predicates[2] as assertions accompanied by terminological structures, as Semantic Networks or Frames, to represent states of the world. A complete world description would be a collection of such predicates. Actions are described by *preconditions* and *effects*[3] and the the agent's intention is described as a *goal*, i.e. a partial world description valid after the plan was excuted.

The nature of the planning processes involved looks two-fold on the surface. One approach is to employ standard search algorithms to solve the planning problem, like *breadth-first search* or informed variants, like *best-first search* or the A^* algorithm [Nilsson, 1971].

These algorithms try to modify the start state by systematically applying legal actions until a state is reached that fulfills the goal description. Yet, the usability of this approach is somewhat limited in the planning domain, due to the great number of states and possible actions [Korf, 1990].

Another approach uses specialized algorithms based on the idea to systematically reduce the differences between start and goal state.

One of the first prominent systems based on this framework is STRIPS [Fikes and Nilsson, 1990][4]. Sacerdoti introduced the idea of generating plans on different levels of abstraction [Sacerdoti, 1990b] and investigated the notion of *nonlinear* planning [Sacerdoti, 1990a][5].

Nonlinear planning introduced a slight shift in the way planning algorithms were looked at. The GPS/STRIPS paradigm worked on states and tried to connect them via actions. Now attention focused on the way actions relate to each other (like destroying each others preconditions), thus making the algorithms to reason about actions rather than states.

The problem of *goal interaction* lead to a large amount of algorithms refining the nonlinear paradigm, most prominent being TWEAK [Chapman, 1990] and adding layers of abstraction as in ABTWEAK [Woods, 1991; Yang *et al.*, 1992].

This planning paradigm faces several problems, like the problem of computational complexity [Bylander, 1991; Gupta and Nau, 1991], stating many interesting problems to unsolvalbe in polynomial time. Another problem — the *frame problem* [Hayes, 1990] — raises

[2] "Predicate instances" seems to be more accurate!

[3] These effects are usually separated into new facts that are valid after the action is completed and obsolete facts that a no longer valid after the action is completed.

[4] According to Nilsson [Nilsson, 1971] STRIPS is a specializiation the GPS approach [Newell and Simon, 1963].

[5] The term "nonlinear" stands for the observation that a plan should not generally contain a fixed linear order of actions, but may allow for actions to executed in parallel or leave the exact order of execution to another system component

the question of correspondance between the internal representation of states and actions and the actual behavior of reality and the executing mechanism in terms of completeness of decriptions. Discrepancies between internal and external situation will eventually lead to severe problems while executing a plan.

3.2 Planning as situated behavior

Classical AI planning methods — as described above — establish a distinct behavior of the executing mechanism: After constructing a plan the resulting sequence of actions is immediately executed under the assumption that the plan is correct. This may well fail under the frame problem or in multi-agent situations where different autonomous agents interact in an unpredictable way.

A more recent framework for the control of an autonomous artificial agent combines planning, execution and sensor data evaluation into architectures that depart from the strict planning/executing behavior.

The following quote is from [Kaelbling, 1990]:

> "There are two problems that arise when a planner is run in a dynamic environment. The first is that, if the planner takes control of the processor, the robot can no longer respond, even at a reflex level, to events in the environment. The second problem is that, during the process of planning, the environment may have changed to such an extent tht the newly created plan is no longer executable in the current situation."

The framework indicated aims at interleaving various processes that take care of the robot's actions on different levels, like emergency or reflex actions (to stop before running into an obstacle), reactive actions (to turn back in front of an obstacle, if this is safe) or planned actions (as resulting from a more classical planner) [Kaelbling, 1990].

The new challenge with this framework is the on of *control structure*, i.e. to devise a connection between these processes that leads to a reasonable behavior of the agent. Numerous approaches have been proposed, cf. [Kaelbling, 1990; Gorgeff and Lansky, 1990; Ambros and Steel, 1990; Flynn and Jones, 1993].

It's interesting to note, that with this approach a strong connection to issues of communication and operating systems — especially real-time operating systems — arises.

4 The scenario

The ICS project aims at the integration of diverse systems, modules and concepts from Artificial Intelligence (AI) reserach into a complex system that is able to solve well-specified tasks that cannot be solved by just one subsystem alone. Therefore, selecting a suitable scenario for the intended system plays a key role. With a trivial scenario, on one hand, reliable results regarding integration cannot be expected; an overly complex scenario, on the other hand, might be unrealistic to realize in the narrow time frame of the ICS project.

Since the authors worked in the field of Planning since more than ten years (e.g. [von Luck and Owsnicki, 1982]) a scenario involving different planning approaches was developed.

4.1 Introductory remarks

As already pointed out in section 3, planning can be viewed from two perspectives, see also [Agre, 1995]:

- Planning as "a relative fixed repertoire of commonly employed structures of action". Resulting systems, like STRIPS or NOAH, employ a hierarchical plan structure whose correctness can be proved – due to the correspondance of internal representation and external world — not counting the frame problem (cf. 3.1, see also [Knoblock, 1994]).
- Planning as a hierarchical process, embodying a set of executable descriptions of an object's behavior. The correctness not be rigoriously proved, since — instead of a formal correspondence — the weaker concept of *convergence* between internal representation and external world is assumed (cf. 3.2).

The situation of the scenario will be a combination of both approaches.

4.2 Scenario details

The *world level* will consist of a *mobile robot* as in [Flynn and Jones, 1993], equipped with a private processor, memory, control program, effectors and sensors. Robots of this type are able to solve easy problems — like traversing a maze or searching objects — using a relatively simple action/monitoring cycle. Their behavior is asymptotic in that they don't follow a fixed structured plan, but improve their current situation towards the goal by evalutating sensor data and following a given program.

More complex planning task, as are solved by classical AI planning systems (cf. [Knoblock, 1994]), assume strict correpondance between representation and world which is not the case with these mobile robots.

Within the ICS project several mobile robots — based on [Flynn and Jones, 1993] — will be designed and tasks of varying complexity will be defined. Tasks reaching the complexity of e.g. SHRDLU [Winograd, 1972] are to our knowledge impossible to execute by a mobile robot of the type described in [Flynn and Jones, 1993]. It's not the primary goal of the

ICS project to improve on the hardware of the robot to enable it to solve the tasks given e.g. in [Winograd, 1972], but to add AI planning methods that have to be modified to widen the usability of the limited robot hardware.

4.2.1 System architecture

As the basic control structure the ICS project will investigate a variant of the *subsumption architecture* from the mobile robots area [Flynn and Jones, 1993]. This architecture can be thought of as a collection of processes with different priorities communicating over a structured *blackboard* [Erman *et al.*, 1980; Hayes-Roth, 1993]. The process with the lowest priority will be able to completely solve the given task employing a very inefficient strategy, if necessary, thus serving as a fall-back strategy. Processes with higher priorities are supposed to solve at least parts of the given task using explicit knowledge and won't be activated if their competence is surpassed.

These processes will be subdivided into relatively independent subtasks embodying the following components (cf. [McDermott, 1993]):

- a control structure for the skeleton of action,
- a knowledge source to decide which subtask is actually active,
- a knowledge source describing different methods to deal with the actual objects.

4.3 Discussion

Integrating both approaches raises a number of problem which have not been satisfactorily solved. Thus this scenario seems to be complex enough to yield realistic results concerning the integration of systems and concepts developed for the respective approaches.

The ICS project is designed to allow statements about whether — and, if yes, how — classical AI planning approaches can indeed enhance the performance of a simple mobile robot. This is done by incrementally raising the difficulty of the tasks posed to the robot thus allowing respective experiments.

We think that this results in a scenario simple enough to sucessfully answer the questions posed in the ICS project, especially since there exists already a promosing proposal for a control structure (cf. 4.2.1).

5 Things to be done

5.1 Research and development tasks

The scenario described above sets up several task to be worked on:

- Evaluation of a platform for knowledge representation that is able to hold knowledge about situations, actions and plans like BACK [von Luck *et al.*, 1987] or MIKE (cf. section 6).
- Evaluation of several linear and non-linear planning systems.
- Control structures for multi-layer planning, like the O-Plan architecture [Currie and Tate, 1990].
- Evaluation of the use temporal representations for planning [McDermott, 1990; Allen, 1990].
- Evaluation of hardware solutions vs. robot simulation testbeds.
- Sub-symbolic modules for adaptive control of actions within (simulated) realistic domains.
- System design/integration issues concerning esp. process communication and data access based on systems like LINDA [Carriero and Gelernter, 1990] (communication) and CORBA (access).

5.2 Research methodology

For the next period of the ICS project we will focus on three major topics:

- Establishing a laboratory with low-budged robots including various testbeds for different task to be fulfilled by these robots. This topic can be seen as the bottom-up part of the ICS project.
- Analysing and experimenting with advanced software components for knowledge representation, planning and monitoring as being listed in section 6. This topic is the top-down part of the ICS project.
- Adapting robot simulation packages from section 6 as a first step into closing the gap between advanced AI planning systems and real world robots. This topic covers the middle out part of the ICS project.

In addition some limit experiments in designing and coding of specialized modules for particular questions will be undertaken. Esp. these experiments will be done not only within the ICS project but as well as part of the practical part of courses of our AI lectures.[6]

6 Resources

This section lists several software packages that are related to the ICS project's objectives. All this software is publicly available and some packages will be evaluated for their utilization within the project.

6.1 Agent-oriented programming

- AGENT0: Implementation of AOP, an interpreter for programs written in AGENT0. AGENT0 is a programming language based on Yoav Shoham's Agent-Oriented Programming Paradigm.

  Implementation: Common Lisp

- ViewGen: ViewGen (Viewpoint Generator) is a Prolog program that implements a "Belief Ascription Algorithm" as described in Ballim and Wilks. This can be seen as a form of agent modeling tool, which allows for the generation of arbitrarily deep nested belief spaces based on the system's own beliefs, and on beliefs that are typically held by groups of agents.

  Implementation: Prolog

6.2 Blackboard architectures

- GBB: UMass GBB is a free blackboard system written in Common Lisp. The commercial system is substantially improved and extended.

  Implementation: Common Lisp

- GEST: GEST (Generic Expert System Tool) is a blackboard system from Georgia Tech.

[6] These limit student project were quite successful in the last half year already.

6.3 Integrated systems

- PAIL: PAIL (Portable AI Lab) is a computing environment containing a collection of state-of-the-art AI tools, examples, and documentation. It is aimed at those involved in AI courses (teaching or learning) at university level or equivalent.

  PAIL is designed around a set of modules, each of which includes a fully implemented and documented computational kernel together with a representative set of autonomous demonstrations and running examples.

  PAIL's current modules cover the following topics

  - Automated Theorem Proving
  - Natural Language Processing
  - Rule-Based Inference
  - Truth Maintenance Systems
  - Constraint Satisfaction Problems
  - Learning
  - Knowledge Acquisition
  - Genetic Algorithms
  - Neural Networks

  Implementation: Common Lisp

- SHRDLU: Winograd's SHRDLU.

  Implementation: MacLisp

- BABYLON: BABYLON is a modular, configurable, hybrid environment for developing expert systems. Its features include objects, rules with forward and backward chaining, logic (Prolog) and constraints.

  Implementation: Common Lisp

- CLIPS: CLIPS 6.0 (C Language Integrated Production System) is an OPS-like forward chaining production system written in ANSI C by NASA. The CLIPS inference engine includes truth maintenance, dynamic rule addition, and customizable conflict resolution strategies. CLIPS includes an object-oriented language called COOL (CLIPS Object-Oriented Language) which is directly integrated with the inference engine.

  Implementation: ANSI C

- MIKE: MIKE (Micro Interpreter for Knowledge Engineering) is a full-featured, free, and portable software environment designed for teaching purposes at the UK's Open

University. It includes forward and backward chaining rules with user-definable conflict resolution strategies, and a frame representation language with inheritance and 'demons' (code triggered by frame access or change), plus user-settable inheritance strategies. Automatic 'how' explanations (proof histories) are provided for rule execution, as are user-specified 'why' explanations.

Implementation: Prolog

- PROTEST: PROTEST is a tool for building prototype expert systems. It accepts a knowledge base written in IF-THEN rules whose conditions and conclusions consist of object/attribute/value triples. Conclusions may contain certainty factors.

  Inference can be either backward chaining or mixed. The former is normal backward chaining, giving a single firm conclusion if one has been deduced or else as a list of possible conclusions in order of their certainties.

  Implementation: Prolog

6.4 Knowledge representation

- COLAB: COLAB is a hybrid knowledge representation system emphasizing the horizontal and vertical compilation of knowledge bases. It has been designed as a COmpilation LABoratory aiming at a synergetic collaboration of different knowledge representation and reasoning formalisms. It is comprised of subsystems dealing with different kinds of knowledge and that can also be used as stand-alone systems. The COLAB representation architecture splits into two main parts, an affirmative part, sometimes also called 'assertional', and a taxonomic part.

  Implementation: Common Lisp

- BACK: BACK supports reasoning based on terminological logics (KL-ONE), supporting inheritance, consistency checking, cycle detection, classification, completion of partial descriptions, role inferences, abox revision, and extended query answering. The main system components are TBox (KB scheme), ABox (KB assertions), IBox (extensional implications), and the uniform access language.

  Implementation: Prolog

- MOTEL: MOTEL is a logic-based knowledge representation language of the KL-ONE family. It contains as a kernel the ALC language which is a decidable sublanguage of first-order predicate logic.

  Implementation: Prolog

- SNePs: SNePS (Semantic Network Processing System) is the implementation of a fully intensional theory of propositional knowledge representation and reasoning.

SNePS includes a module for creating and accessing propositional semantic networks, path-based inference, node-based inference based on SWM (a relevance logic with quantification) that uses natural deduction and can deal with recursive rules, forward, backward and bi-directional inference, nonstandard logical connectives and quantifiers, an assumption based TMS for belief revision (SNeBR), a morphological analyzer and a generalized ATN (GATN) parser for parsing and generating natural language, SNePSLOG, a predicate-logic-style interface to SNePS, XGinseng, an X-based graphics interface for displaying, creating and editing SNePS networks, SNACTor, a preliminary version of the SNePS Acting component, and SNIP 2.2, a new implementation of the SNePS Inference Package that uses rule shadowing and knowledge migration to speed up inference.

Implementation: Common Lisp

6.5 Planning

- 3plan: Three comparable planners – SNLP, TOCL, and TOPI – along with domain and testing information. SNLP is a domain independent systematic conjunctive nonlinear (partial-order) planner that is both complete and sound, TOCL is a total ordered version of SNLP, and TOPI is a backward chaining planner.

  Implementation: Common Lisp

- ABTWEAK: ABTWEAK is a complete hierarchical, non-linear planner that extends David Chapman's (MIT 1986) TWEAK planner as described by Yang (Waterloo) and Tenenberg (Rochester) in 1989. This implementation includes a complete search strategy suited to abstraction hierarchies known as LEFT-WEDGE (Woods 1991). This planner and related work predates that of SNLP.

  Implementation: Common Lisp

- BUILD: A Common Lisp version of Scott Falhman's Build, a blocks-world planner. It is a classical recursive-descent planner, and is one of the first solid geometric modelers. It was converted from CONNIVER to Common Lisp by John Nagle.

  Implementation: Common Lisp

- BURIDAN: This directory contains BURIDAN, an extension of SNLP which breaks free from the tortuous chains of STRIPS-style operators by modeling probabilistic, conditional outcomes. Whereas classical planners' primitive is of the form "Generate a plan that provably achieves goal G", BURIDAN responds to requests of the form "Generate a plan that, with probability at least tau, will achieve goal G".

  Implementation: Common Lisp

- IDM: A STRIPS-like planning system. Examples include standard Blocksworld problems, Towers of Hanoi, and the Register-Swapping Problem.

Implementation: Common Lisp

- NONLIN: An implementation of NONLIN, a hierarchical, non-linear planning system. NONLIN was originally designed and implemented by Austin Tate.

  Implementation: Common Lisp

- Prodigy: An integrated planning and learning system.

  Implementation: Common Lisp

- SOAR: Soar is an integrated agent architecture that supports learning through chunking.

  Implementation: Common Lisp or C

- SPA: SPA is a domain-independent algorithm for case-based plan adaptation that is sound, complete, and systematic. Since SPA is sound, every plan it returns is guaranteed to achieve the goal. Since the algorithm is complete, it is guaranteed to find a plan if one exists, regardless of the initial plan returned from the library. Systematicity guarantees that SPA explores the space of partially specified, incomplete plans avoiding redundant or wasted effort.

  Implementation: Common Lisp

- UCPOP: A domain independent partial order planner that handles an expressive action representation (conditional effects, universally quantified effects, and universally quantified preconditions and goals. UCPOP is both complete and sound.

  Implementation: Common Lisp

- ZENO: A least commitment planner that handles actions occurring over extended intervals of time. ZENO supports deadline goals, metric preconditions, metric effects, and continuous change. It is sound and complete.

  Implementation: Common Lisp

6.6 Robotics

- HYPERBOT: Max Heilperin's simple programmer's interface for robotics sensors and effectors, such as in Lego(R) Technic Control kits and the like, using the Hyperbot(TM) controller.

  Implementation: MIT Scheme

- ARS MAGNA: The ARS MAGNA abstract robot simulator provides an abstract world in which a planner controls a mobile robot. This abstract world is more realistic than typical blocks worlds, in which micro-world simplifying assumptions do not hold. Experiments may be controlled by varying global world parameters, such as perceptual noise, as well as building specific environments in order to exercise particular planner features. The world is also extensible to allow new experimental designs that were not thought of originally.

Implementation: Common Lisp

- Eden: Eden is a Poplog-based AI microworld developed by Simon Perkins, Jocelyn Paine and Edmund Chattoe of the Oxford University Artificial Intelligence Society. It is intended to serve as a testbed for learning and planning programs. Programs written in Pop-11, Prolog, Lisp, and ML control a "bug" that lives in a 2-dimensional world. Any kind of algorithm may be used to control the bug, including genetic algorithms, neural nets, and rule-based systems. Eden consists of a grid of cells, each of which can contain objects such as keys, doors, boulders and quicksand. Bug's objective is to find and eat a piece of food which the simulator has placed somewhere within this grid. To do this, Bug must negotiate its way towards the food while dealing intelligently with obstacles.

  Implementation: Poplog

- Erratic: A mobile robot simulator and controller.

  Implementation: C

- MICE: The Michigan Intelligent Coordination Experiment (MICE) testbed is a tool for experimenting with coordination between intelligent systems under a variety of conditions. MICE simulates a two-dimensional grid-world in which agents may move, communicate, and affect their environment. MICE is essentially a discrete-event simulator that helps control the domain and a graphical representation, but provides relatively few constraints on the form of the domain and the agents' abilities.

  Implementation: Common Lisp

- RSIM: RSIM is a SGI-based graphical robot simulator from the University of Melbourne. It does discrete time simulation of an arbitrary linked robot arm, with full kinematics and dynamics. There is a discrete-time controller and a standard C interface so that users can create and test different controller algorithms.

  The robot simulator currently only works on SGI machines, and in a no-graphics mode on other machines. When the IPRS GUI standard is established, the authors will be converting this to run under IPRS, which will make it machine independent.

  Implementation: C

- Simderella: Simderella is a robot simulator consisting of three programs: CONNEL (the controller), SIMMEL (the robot simulator), and BEMMEL (the X-windows oriented graphics back-end). SIMMEL performs a few matrix multiplications, based on the Denavit Hartenberg method, calculates velocities with the Newton-Euler scheme, and communicates with the other two programs.

  Implementation: C

- TILEWORLD: TILEWORLD is a planning testbed/simulator developed at SRI International by Martha Pollack, Michael Frank and Marc Ringuette.

  Implementation: Common Lisp

- TRUCKWORLD: TRUCKWORLD is a simulated world intended to provide a testbed for AI planning programs, where the planning agent is a truck with arms that roams around the simulated world.

References

[Agre, 1995] Philip E. Agre. Computational research on interaction and agency. *Artificial Intelligence*, 72(1-2):1–52, 1995.

[Allen, 1990] James F. Allen. Towards a general theory of action and time. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 464–479. Morgan Kaufman, 1990.

[Ambros and Steel, 1990] José A. Ambros and Sam Steel. Integrating planning, execution and monitoring. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 735–740. Morgan Kaufman, 1990.

[Bylander, 1991] Tom Bylander. Complexity results for planning. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 274–279, Sydney, Australia, 1991.

[Carriero and Gelernter, 1990] N. Carriero and D. Gelernter. *How to write Parallel Programs*. MIT Press, Cambridge, Mass., 1990.

[Chapman, 1990] David Chapman. Planning for conjunctive goals. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 537–558. Morgan Kaufman, 1990.

[Currie and Tate, 1990] Ken Currie and Austin Tate. O-plan — control in the open planning architecture. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 361–368. Morgan Kaufman, 1990.

[Erman *et al.*, 1980] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):153–167, 1980.

[Fikes and Nilsson, 1990] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 88–97. Morgan Kaufman, 1990.

[Flynn and Jones, 1993] A.M. Flynn and J.L. Jones. *Mobile Robots: Inspiration to Implementation*. AK Peters, Wellesley, MA, 1993.

[Georgeff, 1990] Michael P. Georgeff. Planning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 5–25. Morgan Kaufman, 1990.

[Gorgeff and Lansky, 1990] Michael P. Gorgeff and Amy L. Lansky. Reactive reasoning and planning. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 729–734. Morgan Kaufman, 1990.

[Gupta and Nau, 1991] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 629–633, Anaheim, CA, 1991.

[Hayes-Roth, 1993] Barbara Hayes-Roth. Intelligent control. *Artificial Intelligence*, 59(1-2):213–220, 1993.

[Hayes, 1990] Patrick J. Hayes. The frame problem and related problems in artificial intelligence. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 588–595. Morgan Kaufman, 1990.

[Heinsohn and Owsnicki-Klewe, 1988] Jochen Heinsohn and Bernd Owsnicki-Klewe. Probabilistic inheritance and reasoning in hybrid knowledge representation systems. In W. Hoeppner, editor, *GWAI-88. 12th German Workshop on Artificial Intelligence*, pages 51–60. Springer-Verlag, Berlin, West Germany, 1988.

[Kaelbling, 1990] Leslie P. Kaelbling. An architecture for intelligent reactive systems. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 713–728. Morgan Kaufman, 1990.

[Knoblock, 1994] C.A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.

[Korf, 1990] Richard E. Korf. Planning as search: A quantitative approach. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 566–577. Morgan Kaufman, 1990.

[McDermott, 1990] Drew McDermott. A temporal logic for reasoning about processes and plans. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 436–463. Morgan Kaufman, 1990.

[McDermott, 1993] John McDermott. R1 ("XCON") at age 12: lessons from an elementary school achiever. *Artificial Intelligence*, 59(1-2):241–247, 1993.

[Newell and Simon, 1963] Alan Newell and Herbert A. Simon. Gps: a program that simulates human thought. In Edward A. Feigenbaum and Jerome Feldman, editors, *Computers and Thought*. McGraw-Hill, 1963.

[Nilsson, 1971] Nils J. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, N.Y., 1971.

[Owsnicki-Klewe *et al.*, 1993] Bernd Owsnicki-Klewe, Kai von Luck, and Bernhard Nebel. Wissensrepräsention und Logik — Eine Einführung. In Günther Görz, editor, *Einführung in die künstliche Intelligenz*, pages 3–54. Addison-Wesley, 1993.

[Pylyshyn, 1984] Zenon W. Pylyshyn. *Computation and Cognition: Toward a Foundation for Cognitive Science*. MIT Press, 1984.

[Sacerdoti, 1990a] Earl D. Sacerdoti. The nonlinear nature of plans. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 162–170. Morgan Kaufman, 1990.

[Sacerdoti, 1990b] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 98–108. Morgan Kaufman, 1990.

[Tate *et al.*, 1990] Austin Tate, James Hendler, and Mark Drummond. A review of ai planning techniques. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 26–49. Morgan Kaufman, 1990.

[von Luck and Owsnicki, 1982] Kai von Luck and Bernd Owsnicki. N.N.: A view of planning in chess. In W. Wahlster, editor, *GWAI-82. 6th German Workshop on Artificial Intelligence*, pages 92–101. Springer-Verlag, Berlin, West Germany, 1982.

[von Luck *et al.*, 1987] Kai von Luck, Bernhard Nebel, Christof Peltason, and Albrecht Schmiedel. The anatomy of the BACK system. KIT-Report 41, Department of Computer Science, Technische Universität Berlin, Berlin, West Germany, January 1987.

[von Luck, 1989] Kai von Luck. Repräsentation assertionalen Wissens im BACK-System – eine Fallstudie. KIT-Report 72, Department of Computer Science, Technische Universität Berlin, Berlin, West Germany, May 1989.

[Winograd, 1972] Terry Winograd. *Understanding Natural Language*. Academic Press, New York, N.Y., 1972.

[Woods, 1991] Steven Woods. An implementation and evaluation of a hierarchical nonlinear planner. UW Tech Report CS-91-17, Univ. of Waterloo, 1991.

[Yang *et al.*, 1992] Qiang Yang, Josh Tenenberg, and Steven Woods. Abstraction in nonlinear planning. UW Tech Report CS-92-32, Univ. of Waterloo, 1992.