Planning, Behaviours, Decomposition, and Monitoring Using Graph Grammars and Fuzzy Logic

Jörg-Michael Hasemann
Technical Research Centre of Finland
Machine Automation Laboratory
Mechatronic Systems Group
P.O.Box 13023, 90571 Oulu, Finland
Jorg-Michael.Hasemann@vtt.fi

Abstract

This paper presents new concepts for coordinating multiple behaviours in task level control tasks. Behaviours are introduced as user-defined plans for different tasks with context dependent priorities. A fuzzy logic based Behaviour Control System (BCS) engages and disengages behaviours conditionally at suitable time points, or unconditionally in very urgent cases.

Plans, i.e., behaviours, are user defined context free edge replacement grammars, which introduce parallelisms, loops, and choice in a natural straight forward way. A fuzzy logic based Applicability Control System (ACS) guides the planning process by choosing the "best" applicable production rule. Plan execution and plan validity monitoring are carried out by projecting plan effects using Situation Grids.

Possible applications of this architecture are intelligent systems with high demands for autonomy, flexibility, and problem solving capabilities such as flexible manufacturing cells and mobile robots for forwarding or inspection tasks.

Introduction

Planning systems for real world applications demand context dependent reactivity and behaviour switching, plan execution and plan validation monitoring as well as error handling and plan revision under real time constraints. Hence, they may be better and more precisely denoted as control systems.

For this reason, we believe that a Hierarchical Transition Network (HTN) Planner, as proposed, provides a good framework for real-time operation compared to time-expensive precondition achievement planners despite their proved correctness and completeness under heavy restrictions.

Furthermore, "easy to grasp" representations and engineering tools supporting the design, analysis, and implementation of reactive planning techniques are needed to facilitate their industrial applicability.

Within this paper we propose new concepts for the design of control systems for intelligent systems ranging from teleoperated manipulators to autonomous mobile robots, in essence systems for which the knowledge of operation is known very well beforehand. The key concepts are:

Plans as Graph Grammars. Plans are modeled by context free edge replacement grammars similar to top down hierarchical transition networks. Hence, they feature parallel plan component execution and hierarchical plan decomposition through plan component abstraction. Choice and loops do not exist explicitly but may be introduced using (end-)recursive production rules.

Model Based Behaviours. Behaviours in this approach correspond to the conceptually different high level tasks of an intelligent system and are modeled by plans. See Table 1.

Table 1. Examples for Behaviours with Context Dependent Criticality.

Application	Behaviours employed by the Application
Assembly Robot	different assembly tasks, sensor/tool calibration, active sensing sequences, ...
Planetery Exploration	soil sample collection/delivery, data transmission, data transmission, ...
Shop Floor Robot	deliver, transport, "go out of my"-tasks, recharge batteries, ...

Coordination of Behaviours. Behaviours employ context dependent criticalities, like the need to recharge depends on the current charge level. Hence, depending on the actual situation, behaviour switching may become necessary and must take place at "most suitable" points in time minimizing negative effects of behaviour switching, like performance loss, (mutual) destruction of achieved subgoal etc.

Monitoring Plan Execution and Construction. Obviously, plan execution monitoring, covering all three stages of execution, initial, transient, and goal stage, as well as plan validation monitoring, checking whether the current plan is still valid under the current situation are vitally important.

Fuzzy Logic for Modeling and Reasoning. Fuzzy Logic has been chosen to reason about the criticality and urgency of behaviours and the applicability of production rules during planning. The reason to use it within this work was becuase of the possibility to comprehensively model engineering knowledge ranging from very unsharp, linguistic,

"fuzzy" knowledge up to precise, two-valued, "black and white" knowledge.

Due to space limitations, the concepts of our system can only be sketched within this paper, a more complete thourough description can be found in (Hasemann 1994).

Relation to other Research Efforts

Different aspects of this work have been earlier addressed by other researchers. Examples of hierarchically organized intelligent control architectures (HTN-planner) based on operator abstraction are NASREM (Albus, McCain & Lumia 1989) and PEM (Heikkilä & Röning 1992). Situation abstraction for traditional precondition achievement planners has been employed in ABSTRIPS (Sacerdoti 1974) and ABTWEAK (Yang & Tenenberg 1990).

Behaviours are a metaphor for conceptual activities. However, very often behaviours are associated with the subsumption architecture (Brooks 1986) and consequently became a metaphor for almost direct sensor-actuator coupling denying any representation and conceptualisation. Brooks-like behaviours are low level, biologically oriented mechanisms and are very much the opposite of what is understood as behaviours within this paper. We understand behaviours as high level model based activities represented as plans.

Coordination of Behaviours/Plans, i.e. interrupting and abandoning plans has been earlier addressed by Davis (Davis 1992). However, it remains open how this approach can be extended towards hierarchical abstraction and multiple behaviours.

Monitoring Plan Execution and Validity has surprisingly not been in the focus of research despite its importance for systems operating in real world environments. Plan execution monitoring is usually restricted to pre- and post-condition checking. Transient states are surveyed by a system proposed by (Noreils & Chatila 89) which has been further extended (Chatila, Alami, Degallaix & Laruelle 1992) by hardwiring the different outcomes of a plan component to succeeding plan components. Plan execution monitoring covering all stages of execution (initial, transient, and goal stages) and plan validity monitoring using Situation Grids and World State Aspect Objects has been proposed in (Hasemann & Heikkilä 93).

Fuzzy Logic (Zadeh 1965) for decision support (e.g. Negoita 1985) and process control (e.g. Pedrycz 1989) based on unsharp data and/or rules has been successfully applied in many engineering domains.

Related Control Architectures. Among the large of proposed systems (e.g. Brooks 1986, Kaelbling 1986, Ambros-Ingerson & Steel 1988, Firby 1989), the PRS (Georgeff & Lansky 1987) and the Task Control Architecture (Simmons 1990) are the ones most closely related to the system presented here.

The Task Control Architecture (TCA) is a distributed planning and execution system with centralized control applied to a walking robot. The TCA uses a hierarchical plan representation (TCA Task Tree) with explicit information goals (queries), explicit monitors to check assumptions underlying the plan, and synchronisation constraints to sequentialize plan component execution and delay planning if necessary. The main achievement of this architecture is the development of a framework for parallel execution and planning.

The Procedural Reasoning System (PRS) on the other hand emphasizes the interruptability of "plans" called Knowledge Areas (KA) are defined as sequences of subgoals to be achieved and roughly correspond to behaviours in our approach. KAs are triggered under certain conditions, which in fact corresponds to the invocation of behaviours by the BCS in our approach. Triggered KAs are stacked on the process stack from which one is chosen and pursued. Pursued KAs may be interrupted and replaced with another KA if needed. The PRS, however, lacks monitoring abilities and true coordination of different behaviours due to its unconditional KA switching mechanism.

System Overview

In the following we briefly describe the concepts developed and give a brief overview to the proposed control system architecture.

The proposed system consists of the following components and structures: A set of behaviours which are either active or dormant, a Behaviour Control System (BCS) which continuously determines the criticality of each behaviour and determines possible time points for behaviour switching, a graph grammar representation of each behaviour, an Applicability Control System (ACS) that determines the "best" applicable instantiation for a plan component, and a Situation Grid (SG) used for plan execution and plan validity monitoring.

The design of our control architecture was guided by the believed need to recognise, separate, model, and coordinate conceptually different mutually interrupting tasks for intelligent systems situated in the real world. Considering the tasks of an assembly robot as given in Table 1 illustrates the problem setting. These tasks are very well defined, in the sense that hierarchical plans can be easily formalized for each individual task. Furthermore, tasks may need to be interrupted by other more urgent/critical tasks. E.g. if it becomes necessary to recalibrate the actuator this job might be accomplished at the next non-critical time point. Afterwards the previously running task continues execution.

Obviously, this kind of control system behaviour can be also achieved using traditional control architectures by inserting the planning sequence for recalibration into the plan for assembly together with frequent checking of the trigger conditions. Doing this for two interrupting tasks may work, doing this for a bunch of - possible interrupting - tasks is surely not tractable for a human control system designer; hence our approach.

We claim from the engineering point of view that this - among the lack of tools - is one of the major obstacles towards a broad introduction of intelligent control architectures into industrial applications. We also believe that a behaviour switching mechanism as proposed similar to interrupting mechanisms in operating systems will increase the robustness and the chance of error recovery.

System Components in Details

The control system is described by a set of behaviour each of which is represented by a plan. To each behaviour is attached a measure of criticality which is continuously updated and taken to decide at what points of time behaviours are interrupted. This job is accomplished by the Behaviour Control System (BCS).

Plans

Plans are the central data structure within control systems for intelligent robots as they define the sequence and alternatives, goals are tried to get achieved. Within this paper plans are snapshots of the current planning and execution process continuously changing by the dynamics of planning and execution.

Due to the runtime performance of precondition achievement planners and the low expressiveness of the produced plans, we decided to use the hierarchical top down instantiation scheme of graph grammars as the underlying planning mechanism with an additional control system (ACS) to resolve conflicts among different applicable instantiations.

Obviously, the resulting opportunistic planner is neither complete nor sound for the general case, neither is any other mechanism situated in the real world. The advantage of our planner lies in the expressiveness of the plan (including choice, loop, and hierarchical abstraction), it"s visualization as graphs, the ease of modeling complex behaviours, and last not least it's run time performance, i.e. its applicability for real time tasks.

Within our approach, plans are represented as augmented context-free edge replacement grammars (Habel & Kreowski 1987) over plan components. Abstract plan components or plan frames (pf), to be further decomposed, correspond to nonterminal symbols and atomic plan components (apc), not to be further decomposed, correspond to terminal symbols. The graph grammar is defined as a set of production rules with one nonterminal symbol as the left hand side (lhs) and a double pointed graph (DPG) on the right hand side (rhs) of the production rule. A DPG is defined as a directed graph defining the sequence of execution with a specified entry and exit node to allow embedding of the graph in the parent graph. Within this representation edges correspond to plan components and vertices correspond to (partial) states.

The planning process is then given by successively applying selected production rules to replace nonterminal symbols. The dynamics of planning and execution are given by the availability of planning data and the sequence as described by plan. Completion of execution causes the deletion of the plan component from the current graph. Errors are yet treated by falling back to the next higher level of abstraction and deletion of lower level plan components.

Plan Components

Plan components are the entities of a plan. Three different kinds of plan components may exist:

Plan frames (pf) are conceptual activities corresponding to nonterminal symbols in the graph grammar representation. They are further decomposed during the planning process.

Atomic actions (aa) are the lowest level plan components, e.g. servo commands, corresponding to terminal symbols in the graph grammar representation.

Protection intervals (pi): are virtual, not further decomposed, entities, stating and checking that during its lifetime a specified proposition holds valid. Protection intervals are partly set up automatically by analysing the plan component relations (e.g. single-contributor-single-consumer causal structures) or manually by the control system engineer (e.g. other causal structures).

Plan Component Representation

Plan components are represented as parametrised STRIPS-like operators including initial, transient, and goal state descriptions. Initial state, transient states, and goal state description (add- and delete list) are lists of conjunctively connected predicates. The transient states description describes the behaviour of the plan component qualitatively as well as the constraints put on it, such as security limitations, e.g. maximum speed, minimum distance etc. and is inherited during instantiation.

Furthermore, a plan component state is added to the plan component description providing information about the current state (planned, scheduled, started, ...) of the plan component. Plan component state transitions in turn give rise to different activities like checking the initial, transient, and goal state descriptions.

Situation Grid

The Situation Grid (SG) (Hasemann & Heikkilä 93) is a data structure designed to keep track of the ongoing planning and execution process as well as on relations between each plan component and all others. For monitoring purposes the SG also holds the Initial State, Transient States, and Goal State Descriptions of each plan component as well as the plan component's life cycle state. The relations represented in the SG include:

"Ancestor of (AO)" and "Descendent of (DO)" to describe hierarchical relationships; "Comes before (CB)" and "Comes after (CA)" for sequential relationships, "Optional Choice (OC)" explicit model decomposition choices, i.e. to detect possible imminent conflicts with particular instantiations. Of particular importance is the "No Entry (NE)" relationship denoting that the concerned plan components may be executed concurrently.

The situation grid is updated by messages about newly planned plan component and in turn signals possible or evident conflicts/errors to other system components, like planner/executor. The relationships among plan components within the SG are determined by a small rule base.

The SG grid then continuously checks the currently held assumptions for successful plan execution, similar to verifying the Modal Truth Criterion (Chapman 1985). Within our design, we created one SG for each behaviour and add holding protection intervals from other behaviours as top level activities parallel to all others.

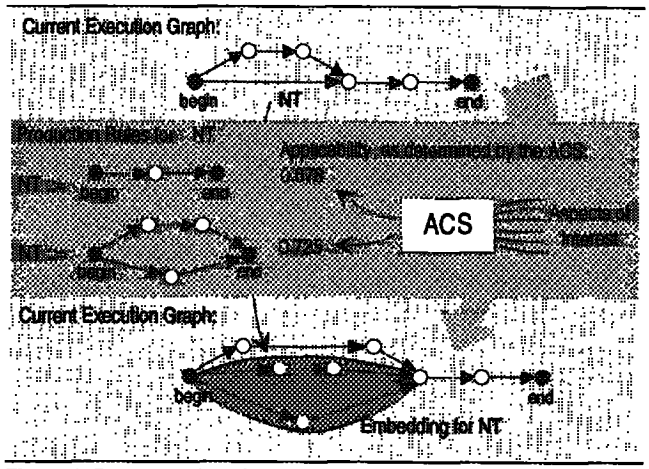

Figure 1. Decomposition of a Plan Frame NT

Applicability Control System

The Applicability Control System (ACS) determines the applicability of each possible instantiation of a particular plan frame and returns (if any) the best applicable instantiation with respect to the current situation. Rules with an applicability below a certain threshold are not considered for application. Fuzzy Logic has been chosen to derive the applicability of a rule and to model the aspects of interest (AOI), which are model as fuzzy variables and determine the different facts constituting the applicability of a rule. Table 2 gives some examples.

Behaviour Control System

Behaviours may be viewed as tasks in multitasking operating systems. Each of the tasks pursues a different goal. Analogously, behaviours correspond to jobs triggered by certain conditions. Consequently, a behaviour switching mechanism is needed to enable and disable behaviours at suitable points in time.

Two methods for behaviour switching have been incorporated: condititonal and unconditional behaviour switching. Whereas conditional behaviour switching includes determining the best possible time for behaviour switching with respect to possibly negative effects on the behaviour to be put asleep, unconditional behaviour switching is more like a reflex producing immediate initiation of the selected behaviour.

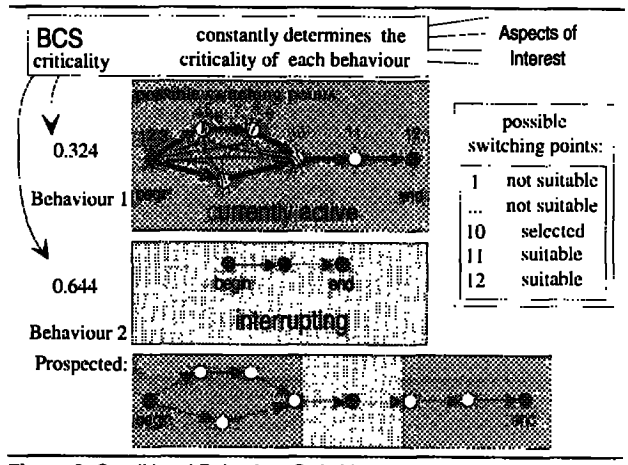

Figure 2. Conditional Behaviour Switching

Conditional behaviour switching is the task of engaging the most critical behaviour with respect to other behaviours. For conditionally interrupting a behaviour, initially the time points of possible interruption are determined, for each of which the set of protection intervals is determined. Among these time points the earliest possible (with no interfering protection intervals) time point is taken. The current protection intervals at the switching point are then added to the SG of the behaviour to be engaged. The disengaged behaviour enters a sleeping mode and waits for the time to wake up. The current limitations of conditional behaviour are still rather strong and hopefully will be relaxed in future: no mutually interrupting behaviours, no plan merging, and no interfering protection interval as-

Table 2. Aspects of Interest for Different Rules

Plan Frame	Aspects of Interest
Grasp Object	object type/size, location, accuracy needed, ...
Move Location	visibility, charge level, ...
Deliver Object	route congestion, receiver location, ...

sumption underlying the conditional behaviour switching mechanism (i.e. no handling of startup and shutdown handling). Admittedly, successive interuption of behaviours may result in an accumulation of global protection intervals and thus cause exclusion of engagement of critical behaviours. However, if the criticality reaches a very high level mutual exclusion is resolved by unconditional behaviour switching at the expense of possible negative effects on other behaviours. This problem is especially apparent if the control system exhibits a high number of behaviours and high frequency switching.

Unconditional behaviour switching occurs when the BCS encounters a criticality above a certain threshold. In this case the currently active behaviour is immediately stopped and enters the sleeping mode. Later when reengaged, the monitoring system automatically detects deviations from the old state and initiates a fall back to a higher level of abstraction in case of error.

Fuzzy Reasoning

Both the ACS and the BCS rely on Fuzzy Logic to model "fuzzy" knowledge and to reason about criticality and applicability. In both cases, aspects of interests (AOI), which themselves may be the result of arbitrarily complex reasoning, are mapped to criticality or applicability.

Each rule has the form: "if AOI X is A then the applicability (criticality) for Y is B", where A and B are fuzzy labels. The fuzzy rules can be represented as tables with respect to each production rule applicable. For a production rule "drive fast" the table may look like this:

Table 3. Fuzzy Rules for a Rule/Behaviour "drive fast"*

drive fast	NB	NM	NS	ZO	PS	PM	PB
visibility	NB	NB	NB	NS	PS	PS	PB
road condition	PB	PS	PS	ZO	NS	NM	NB
load	PB	PB	PS	ZO	NS	NS	NB

* 'NB', ...'ZO',... , 'PB' = (negative big, ..., zero, ..., positive big)

The selection of the AOIs (visibility, road condition, load) as well as all entries in the table are done by the system designer. Each entry in the table denotes the applicability (criticality) of a rule/behaviour wrt. to an AOI value, e.g. "if visibility is NS (negative small) then applicability (criticality) of the drive-fast rule/behaviour is NB (negative big)".

The reasoning process is described as follows: A subscript i $(i=1,...,n)$ distinguishes the n AOIs. The total applicability μ_A is calculated as:

$$\mu_A = \frac{1}{\mu_{A_{max}}} \sum_i \mu_{A_i}(y) \quad (0 \le y \le 1), \text{ with } \mu_{A_{max}} = \max_{0 \le y \le 1} \left[\sum_i \mu_{A_i}(y) \right]$$

and $\mu_{Ai}(y)$ denotes membership value at point y for AOI i. Using the center of gravity method gives a numerical value, interpreted as the applicability (criticality) of the respective rule/behaviour.

Ploughing Field Example

An example, although, due to space limitations, a very simple one, may illustrate the way the control system operates. The scenario is a field that needs to be ploughed. For this a tractor exists which has an enormous fuel consumption (for the sake of this example) so that it needs to be refueled every now then during ploughing the field.

The control system is represented with two behaviours: "plough field" and "refuel" represented by two production rules, one for "plough field" and one "refuel" in Figure 3. All plan components are physical plan components except "plough field" and "refuel" which is defined recursively.

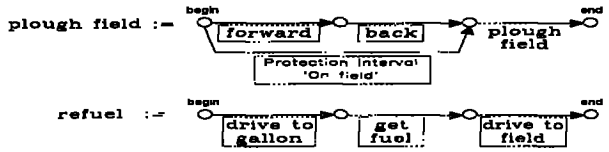

Figure 3. The two Behaviours of the Ploughing Example

The criticality of the two behaviours is determined by the BCS based on the fuzzy rule base attached to each behaviour and is shown in Table 4 and 5. Moreover, if the criticality of a behaviour is extermely low (NB) it is switched off.

Table 4. BCS Fuzzy Rules for Behaviour "plough field"

Behaviour: plough field	NB	NM	NS	ZO	PS	PM	PB
AOI: field ready	PS	PS	PS	PS	PS	PS	NB*

*i.e. behaviour is dormant

Table 5. BCS Fuzzy Rules for Behaviour "refuel"

Behaviour: refuel	NB	NM	NS	ZO	PS	PM	PB
AOI: fuel left	PB**	PB* *	PM*	PM*	NB	NB	NB

* criticality exceeds "plough field" and results in conditionally behaviour switching.

** criticality is extremely high and and causes unconditional behaviour switching

The plan component "plough field" is defined recursively to model a loop, i.e. to plough the field until it is ready. The applicability of the rule "plough field" is determined by the ACS using the rule set given in Table 6. As can be seen termination of ploughing is guaranteed, since "plough field" is no longer applicable when the field is completely ploughed and is behaviour terminates.

Table 6. ACS Fuzzy Rules for Rule "plough field"

Rule: plough field	NB	NM	NS	ZO	PS	PM	PB
AOI: field ready	PS	PS	PS	PS	PS	NB*	NB*

*i.e. rule is not applicable

Table 4 and 5 show the fuzzy rules for the two behaviours "plough field" and "refuel" as used by the BCS. As can be seen the criticality of refuel exceeds the criticality

of "plough field" in case fuel left goes below PS. In case the fuel left value is NM or NB the criticality reaches PB; an indication that an unconditional behaviour switching becomes necessary. In case fuel left is NS or ZO, i.e. still some significant amount of fuel is left, the criticality exceeds the criticality of "plough field" but is still on a non-critical level. Thus conditional behaviour switching is tried. Let's further assume "drive to gallon" and "drive to field" are incompatible with the protection interval "On field". Hence during the time the protection interval is active no behaviour switching can occur. Figure 4 depicts the situation that "forward" is currently execute when conditional behaviour switching becomes necessary. Since possible switching point '1' is within the protection interval "On Field" the earliest point of conditional behaviour switching is '2'.

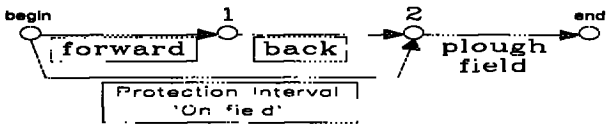

Figure 4. Current Execution Graph. Plan component "forward" is currently under execution.

The criticalities for both behaviours are continuously determined and in case the criticality of "get-fuel" exceeds the criticality of "plough-field" at point "2" behaviour switching will take place at point "2". In case the criticality of "get-fuel" reaches PB during "forward" or "back", unconditional behaviour switching takes place, even if due to "get-fuel" the field needs to be partially re-ploughed again.

Conclusions

Within this paper we presented a new control system for intelligent systems. The system dynamics have been formalised using graph grammars and fuzzy logic to allow flexible and reactive decomposition selection and behaviour switching. The use of fuzzy logic promises easy introduction of linguistic knowledge both for deciding what decomposition to take, and when to switch behaviours and has, to our knowledge, not yet been tried before. Similarly graph grammars provide a solid framework for manipulating graph structures like plans. The system integrates easily in the existing monitoring system based on Situation Grids for plan execution and plan validation monitoring. As of now, behaviour switching is exclusive, i.e. only one behaviour is active at a time. This is an unnatural restriction for non-conflicting behaviours and will be abandoned in future. The introduction of shutdown and bootup activities to store and restore situations may be helpful and is matter of ongoing research. Moreover, further efforts need to be spent to determine the 'best' switching point among all possible. The system described is currently under implementation based on the existing monitoring system. A detailed description can be found in (Hasemann 94).

Acknowledgements

This research was financed by the Technical Research Centre of Finland. The helpful comments of the two anonymous reviewers are also gratefully acknowledged.

References

Albus, J.S.; McCain, H.G.; Lumia, R. 1989. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), NIST Technical Note 1235, 1989 Edition, National Institute of Standards and Technology, U.S. Dept. of Commerce.

Ambros-Ingerson, J.A.; Steel, S. 1988. Integrating Planning and Monitoring, Proc. of the Seventh National Conference on Artificial Intelligence AAAI-88, 83-88. St. Paul, Minnesota.

Brooks, R.A. 1989. A Robot that Walks; Emergent Behaviour from a Carefully Evolved Network, In Proc. of the IEEE International Conference on Robotics and Automation, 693-696. Scottsdale, Arizona.

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence* 32: 333-378.

Chatila, R.; Alami, R.; Degallaix, B.; Laruelle, H. 1992. Integrated Planning and Execution of Autonomous Robot Actions, In Proc. of the IEEE International Conference on Robotics and Automation, Nice, France.

Davis, E. 1992. Semantics for tasks that can be interrupted or abandoned. In Proceedings of the 1st Int'l Conference on AI Planning Systems, 37-43. College Park, Maryland.

Firby, R.J. 1989. Adaptive Execution in Dynamic Domains, Ph.D. diss., Yale University.

Georgeff, M.P.; Lansky, A.L. 1987. Reactive Reasoning and Planning, In Proceedings of the AAAI87, 677-682.

Habel, A.; Kreowski, H.-J. 1987. On context-free Graph Languages generated by Edge Replacement. *Theoretical Computer Science*, Vol. 51, 81-115.

Hasemann, J.-M. 1994. Planning and Monitoring in Dynamic Environments. (Lic. Thesis) Univ. of Oulu, Finland. Forthcoming.

Hasemann J.-M.; Heikkilä T. 1993. A new Approach towards Monitoring in Intelligent Robots, In Proc. of the Scand. Conf. on Artificial Intelligence SCAI'93, 60-76. Amsterdam, Netherlands: IOS Press.

Heikkilä T.; Röning J. 1992. PEM Modelling: A Framework for Designing Intelligent Robot Control, Journal of Robotics and Mechatronics Vol.4 No.5, 432 -444.

Kaelbling L.P. 1986. An Architecture for Intelligent Reactive Systems. Technical Note 400, SRI International, Menlo Park, California.

Negoita C.V. 1985. *Expert System and Fuzzy Systems*. Menlo Park, Cal.: Benjamin/Cumming's Publishing Company, Inc.

Noreils, F.R.; Prajoux R. 1991. From Planning to Execution Monitoring Control for Indoor Mobile Robot, In Proceedings of the IEEE International Conference on Robotics and Automation, 1510-1517. Sacramento, California.

Pedrycz, W. 1989. *Fuzzy Control and Fuzzy Systems*. New York, NY: John Wiley & Sons Inc.

Sacerdoti, E.D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5(2): 115-135.

Simmons, R. 1990. Concurrent Planning and Execution for a Walking Robot. Technical Report, CMU-RI-TR-90-16. Robotics Institute, CMU.

Yang, Q.; Tenenberg, J.D. 1990. ABTWEAK: Abstracting a nonlinear, least commitment planner. In Proceedings of the AAAI-90, 204-209.

Zadeh, L.A. 1965. *Fuzzy Sets*. Information and Control 8, 338-353.