

Lecture Notes in Artificial Intelligence

1772

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Michael Beetz

Concurrent Reactive Plans

Anticipating and Forestalling
Execution Failures



Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Author

Michael Beetz
Universität Bonn
Institut für Informatik III
Römerstr. 164, 53117 Bonn, Germany
E-mail: beetz@cs.uni-bonn.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Beetz, Michael:
Concurrent reactive plans : anticipating and forestalling execution
failures / Michael Beetz. - Berlin ; Heidelberg ; New York ; Barcelona ;
Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1772 : Lecture notes in
artificial intelligence)
ISBN 3-540-67241-9

CR Subject Classification (1991): I.2.9, I.2.11, I.2.8, I.2.6, I.2, C. 2.4, F.3, D.1.3,
D.3.2

ISBN 3-540-67241-9 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP Berlin, Stefan Sossna
Printed on acid-free paper SPIN 10719790 06/3142 5 4 3 2 1 0

Dedicated to the memory of my mother
Renate Beetz

and my grandparents
Elise and Fritz Lehmann

Abstract

Autonomous robots that accomplish their jobs in partly unknown and changing environments often learn important information while carrying out their jobs. To be reliable and efficient, they have to act appropriately in novel situations and respond immediately to unpredicted events. They also have to reconsider their intended course of action when it is likely to have flaws. For example, whenever a robot detects another robot, it should predict that robot's effect on its plan and — if necessary — revise its plan to make it more robust. To accomplish these patterns of activity we equip robots with *structured reactive plans (SRPs)*, concurrent control programs that can not only be interpreted but also reasoned about and manipulated. These plans specify how the robot is to respond to sensory input in order to accomplish its jobs.

In this book we describe a computational model of forestalling common flaws in autonomous robot behavior. To this end, we develop a representation for SRPs in which declarative statements for goals, perceptions, and beliefs make the structure and purpose of SRPs explicit and thereby simplify and speed up reasoning about SRPs and their projections. We have also developed a notation for transforming SRPs, which does not only make the physical effects of plan execution explicit, but also the process of plan interpretation, as well as temporal, causal, and teleological relationships among plan interpretation, the world, and the physical behavior of the robot. Using this notation a planning system can diagnose and forestall common flaws in robot plans that cannot be dealt with in other planning representations. Finally, we have extended the language for writing SRPs with constructs that allow for a flexible integration of planning and execution and thereby turned it into a single high-level language that can handle both planning and execution actions.

Experiments in a simulated world show that by simultaneously forestalling flaws and executing SRPs, the robot can perform its jobs more reliably than, and almost as efficiently as, it could using fixed control programs.

Acknowledgements

This manuscript is a revised version of my dissertation written at the Artificial Intelligence Laboratory at Yale University. In working on this dissertation, I have greatly benefited from the help and support of many people. Here are those I would like to thank especially.

First and foremost, I want to thank my advisor, Drew McDermott. He provided an excellent (intellectual and software) environment in which I could pursue my research. He was always there when I needed advice and gave me the freedom to explore my ideas. During my tenure at Yale I have benefitted immensely from his broad intuitions and his ability to turn vague ideas into working LISP programs.

Ken Yip has been a good friend and mentor throughout. In particular, I want to thank him for teaching me that “knowing” the solution before understanding the problem is not a good way to start one’s research. I hope, I won’t forget too often. Greg Hager and Reid Simmons were the other two members of my Dissertation committee. Their careful reading, critique, and insightful comments have greatly improved this dissertation. I would also like to thank Pam Sturmer and Zach Dodds for taking the time to read and comment on an earlier draft of this dissertation. My gratitude can best be measured by the length of this document. Sean Engelson deserves credit for being a good friend, perfect office mate, and discussion partner.

Since the completion of my thesis I have worked as a member of the RHINO team at the University of Bonn and applied techniques described in this book to problems in autonomous robot control. I would like to thank Prof. Armin B. Cremers for his support and for providing such an excellent research environment. Carrying out successful research in autonomous robot control is not possible without being a member of a team. I had the luck of joining one of the best teams: the RHINO team. I would like to thank especially the following members and alumni: Tom Arbuckle, Thorsten Belker, Wolfram Burgard, Dieter Fox, Henrik Grosskreutz, Dirk Hähnel, Dirk Schulz, Gerhard Lakemeyer, and Sebastian Thrun.

Most importantly, I want to thank my wife Annette and children Nicola and Fabian for their love and support and keeping up with me writing this manuscript.

Table of Contents

Abstract	V
Acknowledgements	IX
List of Figures	XV
1. Introduction	1
1.1 The Approach	3
1.2 Technical Challenges	6
1.3 Introductory Example	8
1.4 Motivation	11
1.4.1 Relevance for Autonomous Robot Control	11
1.4.2 Relevance for AI Planning	12
1.5 The Computational Problem and Its Solution	13
1.5.1 The Computational Problem	13
1.5.2 The Computational Model	14
1.6 Contributions	15
1.7 Outline of the Book	19
2. Reactivity	21
2.1 The DELIVERYWORLD	22
2.1.1 The World	22
2.1.2 Commands and Jobs	25
2.1.3 The Robot	26
2.1.4 Justification of the DELIVERYWORLD	27
2.2 The Implementation of Routine Activities	28
2.2.1 Plan Steps vs. Concurrent Control Processes	29
2.2.2 Interfacing Continuous Control Processes	31
2.2.3 Coordinating Control Processes	33
2.2.4 Synchronization of Concurrent Control Threads	35
2.2.5 Failure Recovery	36
2.2.6 Perception	37
2.2.7 State, Memory, and World Models	37
2.2.8 The Structure of Routine Activities	39

2.3	The Structured Reactive Controller	40
2.3.1	Behavior and Planning Modules	41
2.3.2	The Body of the Structured Reactive Controller	42
2.3.3	Global Fluents, Variables, and the Plan Library	43
2.3.4	The RPL Runtime System	43
2.4	Summary and Discussion	44
3.	Planning	47
3.1	The Structured Reactive Plan	47
3.1.1	Plans as Syntactic Objects	48
3.1.2	RPL as a Plan Language	50
3.2	The Computational Structure	53
3.2.1	The “Criticize-Revise” Cycle	53
3.2.2	The “Criticize” Step	55
3.2.3	The “Revise” Step	66
3.3	The XFRM Planning Framework	66
3.4	Anticipation and Forestalling of Behavior Flaws	67
3.4.1	The Detection of Behavior Flaws	68
3.4.2	Behavior Flaws and Plan Revisions	68
3.4.3	The Diagnosis of Behavior Flaws	69
3.5	Summary and Discussion	72
4.	Transparent Reactive Plans	75
4.1	Declarative Statements	75
4.1.1	RPL Construct Descriptions	78
4.1.2	Achievement Goals	81
4.1.3	Perceptions	83
4.1.4	Beliefs	84
4.1.5	Other Declarative Statements	85
4.1.6	Using Declarative Statements	86
4.2	Routine Plans	88
4.3	The Plan Library	93
4.3.1	Behavior Modules	93
4.3.2	Low-level Plans	93
4.3.3	High-level Plans	95
4.4	Discussion	97
5.	Representing Plan Revisions	99
5.1	Conceptualization	101
5.2	Making Inferences	105
5.2.1	Some Examples	107
5.2.2	Accessing Code Trees	108
5.2.3	Predicates on Plan Interpretations	110
5.2.4	Predicates on Timelines	111
5.2.5	Timelines and Plan Interpretation	112

5.3	Expressing Plan Revisions	114
5.4	XFRML — The Implementation	116
5.5	Discussion	117
6.	Forestalling Behavior Flaws	121
6.1	FAUST	121
6.1.1	The Behavior Critic	121
6.1.2	Detecting Behavior Flaws: Implementation	123
6.1.3	Diagnosing the Causes of Behavior Flaws: Implementation	128
6.1.4	The Bug Class “Behavior-Specification Violation”	134
6.1.5	The Elimination of Behavior Flaws	135
6.2	The Plan Revisions for the Example	135
6.3	Some Behavior Flaws and Their Revisions	140
6.3.1	Perceptual Confusion	141
6.3.2	Missed Deadlines	142
6.4	Summary and Discussion	144
7.	Planning Ongoing Activities	147
7.1	Extending RPL	149
7.1.1	The RUNTIME-PLAN Statement	151
7.1.2	Plan Swapping	154
7.1.3	Making Planning Assumptions	156
7.2	Deliberative Controllers	158
7.2.1	Improving Iterative Plans by Local Planning	158
7.2.2	Plan Execution à la Shakey	158
7.2.3	Execution Monitoring and Replanning	159
7.2.4	Recovering from Execution Failures	160
7.2.5	Some Robot Control Architectures	161
7.3	The Controller in the Experiment	163
7.4	Discussion	164
8.	Evaluation	167
8.1	Analysis of the Problem	167
8.2	Assessment of the Method	170
8.2.1	Description of the Method	170
8.2.2	Evaluation of the Method	170
8.3	Demonstration	176
8.3.1	Evaluating SRCs in Standard Situations	176
8.3.2	Comparing SRCs with the Appropriate Fixed Controller	179
8.3.3	Problems that Require SRCs	181
8.4	Related Work	189
8.4.1	Control Architectures for Competent Physical Agents .	189
8.4.2	Control Languages for Reactive Control	194
8.4.3	Robot Planning	195

9. Conclusion 201

 9.1 What Do Structured Reactive Controllers Do? 201

 9.2 Why Do Structured Reactive Controllers Work? 202

 9.3 Do Structured Reactive Controllers Work for Real Robots? .. 204

References 205

List of Figures

1.1	Top-level view of structured reactive controllers.	3
1.2	Example task in the DELIVERYWORLD.	9
1.3	Comparison between control architectures.	16
1.4	Research directions in planning situated robot control.	17
2.1	The world simulator DELIVERYWORLD.	23
2.2	Implementation scheme for routine activities.	31
2.3	The behavior module LOOK-FOR-PROPS.	32
2.4	Architecture of the structured reactive controller.	41
2.5	Interface of planning modules.	42
3.1	Code tree for a RPL code piece	49
3.2	Replacement of subplan in a code tree.	49
3.3	The “criticize-revise” control strategy.	54
3.4	The “criticize” step.	55
3.5	A projection rule describing the behavior module LOOK-FOR.	60
3.6	An E->P-rule describing the effects of an event.	61
3.7	Task network and a timeline.	63
3.8	Detailed timeline.	64
3.9	Conceptual view of a projected task.	65
3.10	Behavior flaw taxonomy for achievement tasks	70
4.1	Structure of RPL construct descriptions.	79
4.2	detection of flaws in projected robot’s behavior.	81
4.3	The set of RPL plans.	88
4.4	Code tree of a transparent RPL plan	90
4.5	Restartable variants of RPL plans.	92
4.6	Low-level plans for the DELIVERYWORLD.	94
4.7	Low-level plan LOOK-FOR.	94
4.8	Low-level plan for tracking objects	95
4.9	High-level plan for delivering an object	96
4.10	Routine transformation rule.	97
4.11	Entry in HACKER’s plan library.	98

5.1	The XFRML system.	100
5.2	Plan revision rule transforming a code tree.	104
5.3	Graphical representation of a timeline.	112
5.4	Example plan revision rule.	115
5.5	Plan revision rule that transforms more than one subplan.	116
6.1	Algorithm for detecting and diagnosing behavior flaws.	122
6.2	Behavior flaw model for “ACHIEVE-FOR-ALL flaws.”	129
6.3	Specializations of the model “ACHIEVE-FOR-ALL flaw.”	130
6.4	Specializations of the model “clobbered subgoal.”	131
6.5	Specializations of the model “never achieved subgoal.”	132
6.6	Data structure BEH-SPEC-VIOLATION.	134
6.7	Configuration of the structured reactive controller used for solving this problem.	136
6.8	RPL code for the structured reactive controller.	137
6.9	An example execution scenario for the RPL plan in figure 6.8.	138
6.10	Modification of the code tree by the plan revision rule.	139
7.1	Interface of a planning process.	151
7.2	Extraction of a local plan.	153
8.1	Configuration of an SRC.	177
8.2	Performance comparison of different types of robot controllers. ...	180
8.3	Spectrum of robot control problems.	190
8.4	Behavior-based robot control architecture.	191