Modellierung paralleler kontinuierlicher Aktionen von autonomen mobilen Robotern

Henrik Grosskreutz und Michael Beetz
Institut für Informatik III, Universität Bonn,
Römerstr. 164, 53117 Bonn,
email: {grosskre,beetz}@cs.uni-bonn.de

Abstract

Die Fähigkeit von Servicerobotern, das eigene Verhalten beim Ausführen von Plänen vorherzusagen, ist eine wesentliche Grundlage für die Entwicklung von autonomen, ihr eigenes Vorgehen planenden Robotern. Wir beschreiben ein Verfahren zur automatischen Generierung von kausalen Modellen des Verhaltens des mobilen Roboters RHINO bei der Ausführung von Bürobotenaufträgen. Die Modelle werden aus einer sehr kompakten Beschreibung des dynamischen Systems, das Roboter und Umgebung bilden, generiert. Sie berücksichtigen die zeitliche Struktur der kontinuierlichen Prozesse, mögliche wechselseitige Beeinflußung zwischen kontinuierlichen Prozessen, sowie Unsicherheit über den Zustand der Welt. Dies ermöglicht es dem transformationellen Planer XFRM(McD92; BM94), Fehlverhalten wie das Überschreiten von Deadlines vorherzusagen und durch Plantransformation zu vermeiden.

Einleitung

Viele autonome Roboter operieren in nicht vollständig bekannten, sich verändernden Umgebungen. Sie arbeiten dabei nicht mit perfekten Sensoren und Effektoren, und müssen mit nicht vorhersehbaren Veränderungen ihrer Umgebung zurechtkommen. Um trotz unvollständiger Information und nichtdeterministischen Effekten zuverlässig und effektiv zu operieren, müssen die Steuerungssysteme eines Roboters, der in einer solchen Umgebung arbeitet, flexibel auf asynchron eintreffende Sensorinformationen reagieren und kleinere Ausführungsfehler lokal beheben. So muß ein Navigationsprozess auf kleinere Hindernisse reagieren, indem er um diese herumfährt, anstatt mit einem Fehler abzubrechen.

Um solche Steuerungsprogramme (siehe (BBFC98)) effektiv generieren und revidieren zu können müssen KI-Planungssysteme berücksichtigen, daß Roboteraktionen keine deterministischen Effekte haben, daß die Wirkung einer Aktion oft nicht unmittelbar eintritt und daß Prozesse Auswirkungen auf andere, parallel ablaufende Prozesse haben können. Die Verwendung eines Modells, das auf atomaren Aktionen ohne zeitliche Ausdehnung und mit deterministischen Effekten basiert und Pläne als partiell geordnete Menge von atomaren Aktionen darstellt, ist daher für derartige Umgebungen oft nicht adäquat. Modelle, die diesen Eigenschaften Rechnung tragen, müssen zeitliche Strukturen, kontinuierliche Aktionen, exogene Ereignisse, gegenseitige Beeinflussung von Prozesssen, unvollständige Weltbeschreibungen und passive Sensoren berücksichtigen.

Ziel unserer Forschung ist die Modellierung der Steuerung von autonomen mobilen Robotern am Beispiel von RHINO, einem RWI B21 Roboter. Aufgabe von RHINO ist es, in einer Büroumgebung (siehe Abb. 1) als Bürobotenroboter Briefe zwischen verschiedenen Orten zu transportieren. Dabei kann RHINO über EMails sowohl neue Aufträge entgegennehmen, als auch Informationen über Veränderungen der Umgebung (z.B. das Schließen einer Tür) erhalten. Unser Ziel ist es, aussagekräftige Vorhersagen über das Verhalten von RHINO zu erhalten. Mit Methoden des transformationellen Planens kann dann in einem weiteren Schritt, den wir hier nicht behandeln, gegebenenfalls eine Korrektur des Plans vorgenommen werden. Anstatt von atomaren Aktionen auszugehen, verwenden wir in unserem Modell kontinuierliche Prozesse wie Navigation, Positions- und Türöffnungswinkelbestimmung als Basis der Interaktion mit der Umgebung. Diese Prozesse haben eine zeitliche Ausdehnung und können zu einer kontinuierlichen Veränderung von bestimmten Zustandsvariablen, wie z.B. der Roboterposition, führen. Wir unterscheiden dabei zwischen kontinuierlichen Prozessen, die Effektoren - wie z.B. Servomotoren - steuern, und kontinuierlichen Prozessen, die Sensoren auslesen und die gelesenen Daten interpretieren. Diese nennen wir im folgenden Perzeptions-Prozesse. Die kontinuierlichen Prozesse werden durch ein strukturiertes reaktives Programm (SRP) parametrisiert, gestartet und beendet.

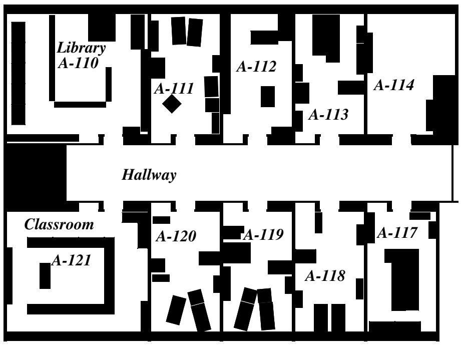

Abb. 1: Die RHINO-Umgebung

Nebenläufige Prozesse

Betrachten wir folgendes Beispiel für eine Situation, in der es zu gegenseitiger Beeinflussung durch nebenläufige Prozesse kommen kann. RHINO erhält den Auftrag, Briefe der Reihe nach zu verteilen. Die Empfänger befinden sich in den Räumen A-118, A-117, A-111 und A-120. Der Brief nach A-111 soll bis 13:30 abgeliefert werden. Zunächst wird ein SRP generiert, bestehend aus Routineplänen zum Anfahren der Räume sowie einem Überwachungsprozess, der prüft, ob die Annahmen über den Zustand der Welt noch gelten. Eine Standardannahme der Routinepläne ist, daß alle Türen offen sind. Ist dem Planungssystem bekannt, daß die Tür zu Raum A-118 geschlossen ist, so wird der Plan modifiziert. Er besteht nun aus drei parallel auszuführenden Unterplänen. Der erste sorgt dafür, daß die Räume A-117, A-111 und A-120 angefahren werden. Der zweite zeigt nur Wirkung, wenn erkannt wird, daß A-118 offen ist. In diesem Fall sorgt er dafür, daß in Raum A-118 gefahren und der entsprechende Brief dort abgelegt wird. Dieser zweite Prozess hat eine höhere Priorität als der erste, er kann den ersten Prozess gegebenenfalls unterbrechen. Zudem sorgt ein dritter Prozess dafür, daß Veränderungen der Öffnungswinkel der nahegelegenen Türen registriert werden. Das SRP des Roboters kann durch folgenden Pseudocode beschrieben werden:

```
execute concurrently
    whenever (near door)
        estimate door angle every 10 seconds
    with-high-priority
        as soon as known (door A-118 open)
            deliver mail to A-118
    execute-in-order
        deliver mail to A-117
        deliver mail to A-111 before deadline 13:30
        deliver mail to A-120
```

Abb. 2 zeigt die Trajektorie, die der Roboter bei der Ausführung des Plans abfährt (durchgängige Linie), und die relevanten Ereignisse, die bei der Projektion durch das Planungssystem mit Hilfe der nachfolgend vorgestellten kausalen Modelle vorhergesagt wurden (Kreise mit Zahlen). Hierbei beginnt RHINO gegen 13:20 mit der Ausführung, und die Tür zu A-118 wird um 13:25 wieder geöffnet. Nachdem RHINO den Raum A-117 (Punkt 2) angefahren hat, steuert er als nächstes A-111 an. Auf dem Weg dorthin stellt er fest, daß die Tür zu A-118 offen ist (Punkt 3), und liefert zuerst den Brief nach A-118 ab (Punkt 4), da dieser eine höhere Priorität hat. Erst danach wird der Brief an A-111 abgeliefert (Punkt 5). Dies führt zum Überschreiten der Deadline. Dieses Fehlverhalten sollte das Planungssystem prognostizieren, wenn es die Information erhält, daß die Tür zu A-118 demnächst geöffnet wird.

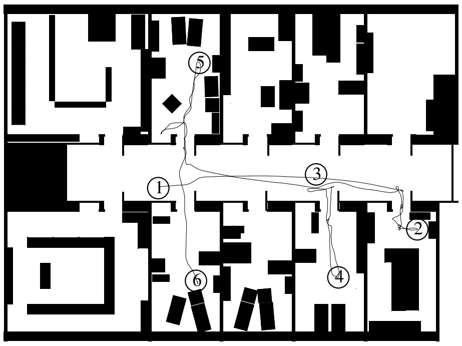

Abb. 2: Die Trajektorie des Roboters

Wie wir aus dem Bild erkennen können, stimmt die symbolische Vorhersage und die Ausführung stark überein. Ferner enthält die symbolische Projektion alle Ereignisse, wie das Passieren von Türen, verlassen von Räumen etc. die auf die Ausführung des Plans einen Einfluß haben.

Ein anderes Beispiel ist der Erhalt der Nachricht: “Es steht ein Tisch hinter der Tür zu A-120”. Dies sollte dazu führen, daß ein möglicher Zusammenstoß mit dem Tisch vorhergesagt wird, weil RHINO seine Sonarsensoren vor Türen abschaltet. Um derartiges vorherzusagen, muß ein Modell des Systems kontinuierliche Aktionen, exogene Ereignisse, gegenseitige Beeinflussung von Prozesssen, und passive Sensoren beinhalten.

Modellierungsprinzipien

Das Verhalten des Roboters, seiner Kontrollprogramme und der Umgebung kann modelliert werden als ein Kontroll-Prozess, der ein dynamisches System, die Umgebung, kontrolliert (siehe Abb. 3). Der Zustand des Systems wird von Perzeptions-Prozessen über verschiedene Sensoren geschätzt. Der Roboter kann über seine Effektoren Zustandsvariablen verändern. Die Kommunikation zwischen Perzeptions-Prozessen und SRP geschieht über sogenannte Fluenten. Dies sind Variablen, die eine Veränderung ihres Wertes signalisieren, und dadurch zur Prozesssynchronisation und -kommunikation verwendet werden können.

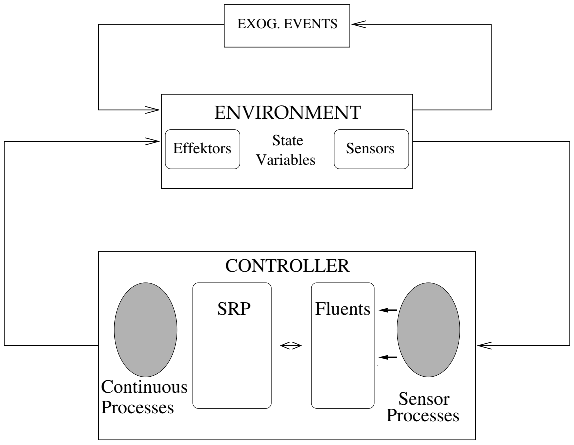

Abb. 3: Das allgemeine Prozess-Modell

Damit der Planer in der Lage ist, Fehlverhalten des Roboters vorherzusehen und rechtzeitig Korrekturen am Plan vorzunehmen, müssen die Vorhersagen über das Verhalten des Roboters effizient und kompakt generiert werden. Die Kompaktheit impliziert, daß nur die wesentlichen Ereignisse vorhergesehen werden, um den Suchraum nicht unnötig zu vergrössern, den ein Planer, der diese Vorhersage erhält, betrachtet. Damit ist es insbesondere bei kontinuierlichen Prozessen nicht möglich, die Auswirkungen zu jedem Zeitpunkt zu vermerken. Dennoch kann nicht ohne weiteres ein einfacheres Modell der Umgebung verwendet werden, da sonst die Wechselwirkung zwischen den Prozessen nicht mehr erfaßt würde. Wir machen daher einige Annahmen über das System sowie über den Roboter-Controller, die es uns ermöglichen nur wenige relevante, qualitative Veränderungen zu projizieren.

1. Die Welt wird außer durch die Effektoren des Roboters nur durch das Eintreten bekannter exogener Ereignisse verändert. Wir betrachten bislang nur Klassen von nicht-kontinuierlichen exogenen Ereignissen.

2. Kontinuierliche Veränderungen, die die Zustandsvariablen SV_1 bis SV_n verändern, induzieren eine stetige Kurve im Raum $\bigotimes_{i=1}^{n} SV_i$. Diese Kurve kann approximiert werden durch einen Polygonzug. Die Eckpunkte dieses Polygonzugs nennen wir $steps$.

3. Eine kontinuierliche Veränderung führt dann und nur dann zu einer Veränderung der Konfiguration der kontinuierlichen Prozesse, wenn während des Übergangs von $step_i$ zu $step_{i+1}$ Fluenten getriggert werden.

4. Alle Veränderungen des Systems, auf die das SRP reagiert, werden von Sensor-Prozessen festgestellt und dem SRP über Fluenten mitgeteilt.

Die Bedeutung dieser Annahmen wollen wir nochmals kurz hervorheben:

- Die Synchronisation von Prozessen darf nur über Fluenten ablaufen. Ein aktives Warten auf eine Bedingung wird nicht berücksichtigt.

- Wir modellieren nicht sowohl die Sensoren als auch die davon abhängenden Perzeptions-Prozesse jeweils getrennt, sondern verwenden nur ein Modell, das die Fluent-Ausgabe des logischen Sensors in Abhängigkeit der Zustandsvariablen beschreibt.

Unter diesen Annahmen ist es nun möglich, das System und den Controller auf einem recht hohem Abstraktionsniveau zu spezifizieren. Dazu sind folgende Angaben nötig:

1. Die Menge der Zustandsvariablen des Systems.

2. Zu jedem Perzeptionsprozess die gemessenen Zustandsvariablen, die Fluenten die der Prozess verändert sowie ein Sensormodell, daß die Schätzfunktion von den Zustandsvariablen auf die Fluenten sowie die Umkehrabbildung der Schätzfunktion beinhaltet.

3. Zu jedem kontinuierlichem Prozess die Menge der veränderten Zustandsvariablen, sowie eine Funktion, die eine Parametrisierung des Prozesses auf einen Polygonzug im Raum der Zustandsvariablen abbildet.

Modell des RHINO-Systems

Betrachten wir ein konkretes Beispiel. Wir wollen einen Teil des Navigationssystems eines Büroboten-roboters beschreiben, das auf RHINO realisiert ist. Das System besitzt einen Kontrollprozess zur Navigation. Diesem wird eine Folge von Zielpunkten übergeben, wobei der letzte Punkt das tatsächliche Ziel ist, während die anderen Punkte lediglich Zwischenpunkte sind. Die Navigation zwischen diesen Punkten erfolgt mit Hilfe eines Pfadplaners (TBB+98). Das System benutzt ein Verfahren zur passiven Schätzung der Roboterposition in der Umgebung (BFHS96; BFH97). Außerdem besitzt es einen probabilistischen Schätzer für den Öffnungswinkel von Türen, der immer dann aktiviert wird, wenn RHINO an einer Tür vorbeikommt. Das Ziel ist, ein kausales Modell zu generieren, das qualitativ vorhersagen kann was passiert, wenn der Roboter Pläne wie den obigen ausführt. Das konkrete Systemmodell für dieses Steuerungssystem zeigt Abb. 4.

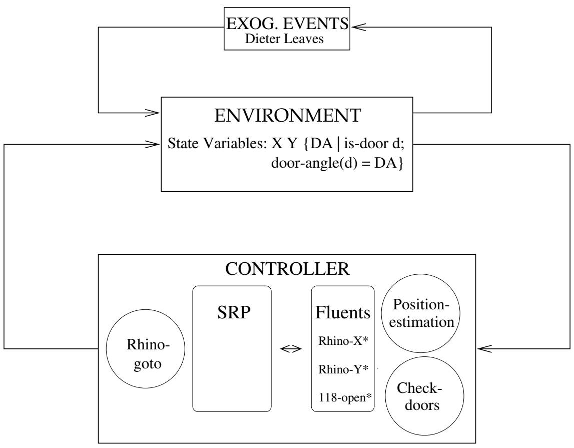

Abb. 4: Das konkrete Prozess-Modell

Unser Planungssystem erwartet eine Spezifikation des dynamischen Systems sowie der verschiedenen zugehörigen Prozesse. Diese erfolgt in diesem Beispiel folgendermaßen:

```
(def-dynamical-system Rhino-Umgebung
    (:state-vars (x "X-Position des Roboters")
                 (y "Y-Position des Roboters")
                 (:static map "Karte der Umgebung")
                 (:initializer rhino-sv-init))
    (:state-functions
                 (door-angle "Tür-Öffnungswinkel")
                 (:initializer rhino-sf-init))
    (:sensor-processes
                 position-estimation
                 door-angle-estimation)
    (:control-processes rhino-goto)
    (:exogeneous-events person-arrives))
```

Die Spezifikation enthält die Systemvariablen sowie Angaben welche kontinuierlichen Prozesse und Perzeptionsprozesse das SRP verwendet. Um die Angaben zum System möglichst kompakt zu halten, verwenden wir zudem *state-functions*. Dies sind Funktionen, die Bezeichner, in unserer Umgebung beispielsweise Türnamen, auf nicht direkt benannte Zustandsvariablen abbilden. Außerdem wird der Wert der Zustandsvariablen zu Beginn der Projektion über bestimmte Funktionen, genannt *initializer*, berechnet.

Neben dem Gesamtsystem müssen noch die einzelnen Prozesse spezifiziert werden. Der Steuerprozess RHINO-GOTO wird wie folgt spezifiziert:

```
(def-continuous-process rhino-goto (dest-descr)
    (:modifies x y)
    (:polyline compute-goto-steps
          "Abb: Zielbeschreibung → Trajektorie")
    (:start-esc start-nav)
    (:stop-esc stop-nav))
```

Der Prozess RHINO-GOTO erhält als Argument eine Beschreibung des anzufahrenden Ziels. Zum Erreichen dieses Ziels führt der Prozess eine kontinuierliche Veränderung der Zustandsvariablen x und y durch. Den Polygonzug, den diese Veränderung im Raum $x \times y$ beschreibt, liefert die Funktion compute-goto-steps, parametrisiert durch die Zielbeschreibung. Zudem werden zu Beginn und am Ende des Prozesses die Funktionen start-nav und stop-nav wegen ihrer Seiteneffekte aufgerufen.

Diese Spezifikation wird in eine Menge von ca. 12 Projektionsregeln (siehe (McD94)) transformiert, die dafür sorgen, daß nur an den Eckpunkten (bzw. $steps$) des Polygonzugs Ereignisse generiert werden. Diese führen eine Aktualisierungen der Fluenten auf der Grundlage der neuen Werte der Zustandsvariablen durch. Zwischen solchen $steps$ wird lediglich die Ableitung der Zustandsvariablen nach der Zeit vermerkt, so daß deren zwischenzeitlicher Wert gegebenenfalls rekonstruiert werden kann.

Perzeptionsprozesse wie der Positionsschätzprozess POSITION-ESTIMATION werden folgendermaßend spezifiziert:

```
(def-sensor-process position-estimation
    (:output-fluents (rhino-x* measures x)
                     (rhino-y* measures y))
    (:sensor-estimation loc-estimation)
    (:sensor-trigger loc-trigger))
```

POSITION-ESTIMATION schätzt die Zustandsvariablen x und y, und liefert seine Ergebnisse über die Fluenten RHINO-X* und RHINO-Y* an das SRP. Die tatsächliche Schätzfunktion, angewendet auf die evtl. fehlerhaften Sensordaten, wird durch die Funktion loc-estimation approximiert. loc-trigger ist die Umkehrkunktion von loc-estimation.

Auch hier wird die Spezifikation in eine Menge von Projektionsregeln transformiert (ca. 20). Diese führen im wesentlichen zu folgendem Verhalten:

1. Immer wenn Threads blockieren, weil sie auf einen Fluenten warten, werden die Bedingungen des Fluenten an die Zustandsvariablen extrahiert. Dazu wird zunächst Bestimmt, von welchen Zustandsvariablen $sv_1, ..., sv_m$ der Fluent abhängt. Dann werden diejenigen Regionen im Raum $\bigotimes_{i=1}^{m} SV_i$ berechnet, in denen der Fluent seinen Wert ändert. Diese werden als *trigger-areas* vermerkt.

2. Vor jeder kontinuierlichen Veränderung von $step_i$ zu $step_{i+1}$ wird bestimmt, ob es zu einem Schnitt von einer *trigger-area* mit der durch die $steps$ induzierten Strecke durch $\bigotimes_{i=1}^{m} SV_i$ kommt. Wenn ja, so wird am Schnittpunkt ein *passive-sensor-update* Ereignis generiert. Dieses sorgt dafür, daß die Fluenten aktualisiert werden, was zur Befreiung von blockierten Threads führt.

Die so generierten Projektionsregeln bilden dynamisch in Abhängigkeit der Prozess-Konfiguration ein kausales Modell der kontinuierlichen Prozesse. Das kausale Modell beachtet nur die Zeitpunkte, an denen sich die Ableitung der Zustandsvariablen ändert, oder an denen die Prozesskonfiguration sich durch Blockieren oder Befreien von Threads verändern kann. Die Zeiträume zwischen solchen Punkten werden übersprungen. Diese Modelle liefern damit Vorhersagen, die nur qualitative Veränderungen des Systems und des Controllers liefern, und somit relativ kompakt bleiben.

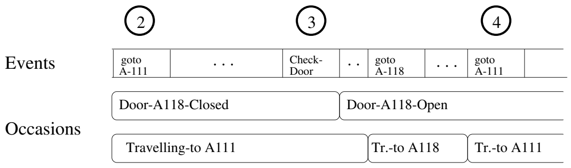

Abb. 5: Die Timeline

Realisierung und Evaluierung

Unsere Modelle bauen wir im Rahmen des transformationellen Planers XFRM(McD92; McD94) auf. Dieser bietet eine sehr mächtige Modellierungssprache für dynamische, unvollständig bekannte Umgebungen. Die SRP sind in RPL (McD91) geschrieben, einer Lisp-ähnlichen Roboter-Kontroll-Sprache, die Schleifen, bedingtes Verhalten, Variablen, prozedurale Abstraktion, parallele Prozesse sowie Synchronisationsmethoden zwischen diesen bereitstellt.

Unser Ansatz ermöglicht es, das dynamische System, das Roboter und Umgebung bilden, auf einer recht hohen Abstraktionsebene zu beschreiben. Ausgehend von dieser Beschreibung werden kontextabhängige XFRM-Modelle des Systems generiert (BG98). Diese berücksichtigen Unsicherheit im Weltwissen, kontinuierliche Veränderung des Systems durch kontinuierliche Prozesse, passive Sensoren und die Wechselwirkungen zwischen Perzeptions-Prozessen, SRP's und kontinuierlichen Prozessen.

Diese Modelle ermöglichen es, Projektionen der SRP zu generieren die wesentliche Ereignisse, die bei der Ausführung des SRP eintreten, vorhersehen (siehe Abb. 5). Die Projektionen bilden eine ausreichend genaue Vorhersage des Verhaltens des Systems um Fehler wie Deadlineüberschreitungen oder Zusammenstöße mit Hindernissen nach dem Abschalten der Sonarsensoren vorhersagen zu können, und ermöglichen es damit dem Planungssystem, diese Fehler zu beheben (BM94). Uns sind keine anderen KI-Planungsysteme bekannt, die diese Aussagekraft haben. Unsere weitere Forschung im Bereich der Modellierung autonomer mobiler Roboter wird sich auf die Verallgemeinerung der hier vorgestellten Ansätze, die Modellierung weiterer Prozesse sowie das Lernen von Parametern des Systems konzentrieren.

References

M. Beetz, W. Burgard, D. Fox, and A. Cremers. Integrating active localization into high-level control systems. *Robo-*

tics and Autonomous Systems, 1998.

Wolfram Burgard, Dieter Fox, and Daniel Hennig. Fast grid-based position tracking for mobile robots. In *Proceedings of the 21th German Conference on Artificial Intelligence (KI 97), Freiburg, Germany*. Springer Verlag, 1997.

Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Position tracking with position probability grids. In *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROMICRO '96)*, pages 2–9. IEEE Computer Society Press, 1996.

M. Beetz and H. Grosskreutz. Causal models of mobile service robot behavior. In R. Simmons, M. Veloso, and S. Smith, editors, *to appear in Fourth International Conference on AI Planning Systems*, Morgan Kaufmann, 1998.

M. Beetz and D. McDermott. Improving robot plans during their execution. In Kris Hammond, editor, *Second International Conference on AI Planning Systems*, pages 3–12, Morgan Kaufmann, 1994.

D. McDermott. A reactive plan language. Research Report YALEU/DCS/RR-864, Yale University, 1991.

D. McDermott. Transformational planning of reactive behavior. Research Report YALEU/DCS/RR-941, Yale University, 1992.

D. McDermott. An algorithm for probabilistic, totally-ordered temporal projection. Research Report YALEU/DCS/RR-1014, Yale University, 1994.

S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998. to appear.