

Oberseminar Künstliche Intelligenz

Schauinsland 6.-7.10.2004

Jussi Rintanen & Stefan Wöfl, editors

Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, 79110 Freiburg im Breisgau
Germany

Foreword

These are the abstracts of the Foundations of Artificial Intelligence laboratory seminar that took place at the University of Freiburg house on the Schauinsland mountain on the 6th and 7th of October 2004.

Contents

Foreword	ii
Table of Contents	iii
Contributions	1
Dr. Sven Behnke: <i>Humanoide Roboter</i>	1
Michael Brenner: <i>Continuous planning in dynamic partially-observable environments</i>	3
Dr. Markus Büttner: <i>Application of partial-order reduction in planning</i>	6
Mehmet Giritli: <i>Logics of vector spaces</i>	9
Malte Helmert: <i>Dialectic strategy search: Hegel meets Mehdorn</i>	12
Alexander Kleiner: <i>Active search and rescue within simulated disaster areas</i>	15
Christian Köhler: <i>Qualitative spatio-temporal queries on trajectory data from model-based tracking</i>	16
Sebastian Kupferschmid: <i>An introduction to timed automata and model checking</i>	19
Jürgen Müller: <i>Learning by imitation</i>	22
Dr. Jussi Rintanen: <i>New developments in planning as satisfiability</i>	24
Alexander Scivos: <i>Double crossing is considered harmful</i>	27
Dr. Jan-Georg Smaus: <i>Termination of logic programs using various dynamic selection rules</i> . . .	28
Thilo Weigel: <i>KiRo - A table soccer robot ready for the market</i>	29
Dr. Stefan Wölfl and Marco Ragni: <i>Branching Allen - Reasoning with intervals in branching time</i>	32

Dr. Sven Behnke

Humanoide Roboter

Humanoide Roboter haben in letzter Zeit viel Aufmerksamkeit in den Medien und der Forschung erhalten. Solche Roboter haben eine menschenähnliche Körperform, ähneln in der sensorischen Ausstattung dem menschlichen Vorbild und haben vergleichbare Aktionsmöglichkeiten. Damit humanoide Roboter erfolgreich in einer komplexen Welt agieren können, müssen sie ihr Verhalten an die jeweilige Situation adaptieren können, müssen lernen und sich entwickeln.

In meinem Vortrag möchte ich zunächst motivieren, warum humanoide Roboter ein interessanter Forschungsgegenstand sind. Dabei spielt einerseits der praktische Aspekt eine Rolle. Eine menschenähnliche Körperform ist ideal für einen Roboter, der in einer für Menschen gestalteten Umgebung agieren soll. Treppen, Türklinken, Schalter, Werkzeuge, usw. sind den menschlichen Proportionen angepasst. Markierungen und Beschriftungen sowie akustische Signale sprechen das menschliche visuelle und auditorische System an. Die Vorteile humanoider Roboter kommen insbesondere in der direkten Interaktion mit Menschen zum Tragen. Die Kommunikation via Sprache, Blick, Mimik, Gestik und Körperhaltung wird in der zwischenmenschlichen Interaktion von früher Kindheit an geübt. Sie stellt deshalb eine ideale intuitive Benutzerschnittstelle für die Mensch-Maschine-Kommunikation dar. Die dem menschlichen Vorbild nachempfundenen Bewegungsabläufe humanoider Roboter machen durch ihre Vorhersagbarkeit eine Zusammenarbeit mit Menschen auf der Grundlage nonverbaler Kommunikation möglich.

Neben dem profanen Aspekt der Nützlichkeit humanoider Roboter gibt es auch eine Motivation aus der Künstlichen-Intelligenz-Forschung (KI) zum Bau humanoider Roboter. Eine wesentliche Methode der KI-Forschung ist der konstruktive Ansatz. Durch Schaffung intelligenter Artefakte wird versucht, Intelligenz zu verstehen. Dabei hat sich die Erkenntnis durchgesetzt, dass Intelligenz nicht ohne einen Körper existieren kann (Embodiment) und nur durch Interaktion mit der Umwelt sichtbar wird (Situativeness). Der Bau von Robotern, die in komplexen Umgebungen agieren, dient dabei der Verankerung von Symbolen in der Welt, ohne die diese bedeutungslos sind. Roboter dienen auch zum Testen von Prinzipien, die durch Abstraktion biologischer Modelle gewonnen werden, z.B. zum Navigationsverhalten von Wüstenameisen. Folgt man diesem Ansatz zur Erforschung menschlicher Intelligenz, so muss man den Bau humanoider Roboter in Angriff nehmen.

In meinem Vortrag werde ich einige Projekte zum Bau humanoider Roboter vorstellen. Dabei werde ich insbesondere auf Laufroboter, wie Asimo und Johnnie, Manipulationsroboter, wie Wendy und Armar, sowie Kommunikationsroboter, wie Kismet und Mexi eingehen.

Danach werde ich über einige Aktivitäten aus unserem Projekt "Lernende Humanoide Roboter" berichten. Ich werde den im Bau befindlichen Roboter Alpha vorstellen und dabei auf das mechanische Design, die Elektronik, Kommunikation, Perzeption, Verhaltenssteuerung und Simulation eingehen. Des Weiteren werde ich über den Umbau eines kommerziellen humanoiden Roboters, Robo-Sapien, zu einem programmierbaren autonomen Roboter mit visueller Wahrnehmung berichten.

Da humanoide Roboter vielfältige Aufgaben erfüllen sollen, ist deren Evaluierung nicht einfach. Eine mögliche Methode ist die Ausrichtung von Wettbewerben, z.B. im Roboterfußball. Um einen Eindruck vom Stand der Entwicklung zu vermitteln, werde ich über die Liga der humanoiden Roboter im diesjährigen RoboCup berichten. Dabei wird offensichtlich werden, dass noch viel Forschungsarbeit zu leisten ist, damit humanoide Roboter sinnvoll eingesetzt werden können.

Zu den aktuellen Forschungsfragen gehören die Entwicklung leistungsfähigerer Aktuatoren und Sensoren sowie die autonome Energieversorgung über längere Zeit. Eine besondere Herausforderung ist die Realisierung dynamischen Laufens, welches die Systemdynamik unterstützt und somit energiesparend ist. Da humanoide Roboter über viele Freiheitsgrade verfügen, sind modulare Architekturen nötig, um die Kontroll-

probleme handhabbar zu machen. Auch sind Konzepte zur Integration von Einzelkomponenten nötig, damit sich Synergieeffekte, wie z.B. bei der audio-visuellen Spracherkennung oder der aktiven Wahrnehmung, ergeben können. Nicht zuletzt sind Lernverfahren erforderlich, um das Verhalten humanoider Roboter an die Erfordernisse der jeweiligen Situation anzupassen. Dabei ist es nötig, Verfahren zu entwickeln, die aus wenigen Beispielen lernen können. Insbesondere das Imitationslernen bietet eine Möglichkeit, humanoide Roboter auch durch naive Benutzer programmieren zu lassen.

Am Ende meines Vortrags werde ich einen Ausblick auf mögliche Entwicklungsszenarien im Bereich humanoider Roboter geben. Insbesondere der Einsatz als "Persönlicher Roboter" bietet dabei interessante Perspektiven.

Michael Brenner

Continuous planning in dynamic partially-observable environments

The EU-funded project *CoSy* (starting this month) has the visionary objective of constructing

“physically instantiated [...] systems that can perceive, understand [...] and interact with their environment, and evolve in order to achieve human-like performance in activities requiring context-(situation and task) specific knowledge.”

The project is an effort to bring together the results of 20 years of increasingly specialized research in increasingly fragmented subfields of AI and Cognitive Science, and to assess possibilities of their integration in complex autonomous embodied systems. Apart from theoretical interests it is the concrete goal of the project to build *“a robot with many of the capabilities of a typical human 4-5 year old child”!*

Our department and Prof. Burgard’s will participate in the project, mostly in a work package named *Planning and Failure Detection* concerned with planning, plan execution, and monitoring, reasoning about actions and plans, extending one’s capabilities, and cooperation with other agents. This talk presents some of my ideas about planning in an embodied agent at project start.

Planning and acting in realistic environments poses a number of difficulties to artificial agents many of which are due to the dynamic nature and partial observability of the real world: other agents’ actions as well as naturally occurring events (e.g. changing of light conditions) may affect the agent’s surroundings in ways she cannot foresee, control or even perceive. It is thus crucial that agents are able to act despite lack of information at planning time and that during plan execution they are able to react to changes in the world (e.g. stop moving when there is a human standing in your way).

Unfortunately, this leads to a problem. The more dynamic the world the more error-prone an agent’s plans will be (if it is possible to find plans at all). However, it is also not possible to give up planning at all or to use purely reactive forms of planning because complex problems may only be solvable deliberately and in a distinctly goal-driven way. Indeed, the answer to lack of knowledge and dynamics must not be to abandon planning but to make it more flexible and to make agents even more proactive so that they can actively extend and update their knowledge when necessary.

Therefore contingencies and lack of knowledge have to be taken into account during planning. Traditionally, research in AI Planning has addressed this problem in the subfields of Conditional Planning and Probabilistic planning. The problem solved by conditional and probabilistic planning algorithms is to find plans that will work under all circumstances imaginable, a fact that makes the problem computationally hard. Considering the large number of unobservable features and possible contingencies in dynamic multiagent environments it is clear that even small-sized problems will be hard to solve by conditional or probabilistic planners.

Luckily, in realistic domains like those considered in the *CoSy* project there is often no need to devise universally executable plans before acting. Since agents are *continuously* planning, acting, evaluating the results of their actions, and learning from it, they may deliberately choose to postpone the resolution of contingencies and handle them only later when more information has been gained.

Interleaving planning and execution in this manner requires some amount of introspection from the agents on the planning process itself, since an agent must decide when to stop planning and start executing a plan fragment (and when to resume planning again). To this end we will introduce a modelling technique that allows agents to reason directly about their knowledge during planning and about the knowledge they may gain during execution.

To model the epistemic nature of “states” reasoned about during planning we extend the domain of each (multi-valued) state variable with a value `unknown`. Usually, unknown state variable values are not updated

by actions but by *perceptions*. To model different kinds of sensor capabilities the conditions under which specific perceptions are made must be explicitly described by *perception rules* and *sensing actions*. Below is an example of a perception rule describing that once an agent is in a room she will (automatically) perceive the light conditions in the room. Note that `(loc ?a : ?r)` describes a state variable `(loc ?a)` with value `?r`. As “effect” of the rule no valuation of the state variable `(lightcondition ?r)` is given, since it cannot be known at planning time.

```
(:perception PERCEIVE-LIGHT-CONDITION
  :agent ?a
  :parameters ?r - room
  :precondition (loc ?a : ?r)
  :perceive (light-condition ?r)
)
```

Perception rules allow to achieve *knowledge goals*, i.e. knowing the time. Most often, however, knowledge is a precondition for further actions and/or planning. Since the actual values will only be known at execution time, but the (goal-driven) agent needs a full plan much earlier, we introduce the concept of *assertions*. Assertions are placeholders for yet unspecified parts of a plan but some conditions and effects of which are already known. A simple example is given below.

```
(:assertion ASSERT-LIGHTS-ON
  :agent ?a
  :parameters ?a - agent ?r - room
  :trigger (light-condition ?r)
  :precondition (loc ?a : ?r)
  :assert (light-condition ?r : ON)
)
```

Agents can use assertions just like actions in their plans (preconditions and the “trigger” knowledge condition must be satisfied before). Yet, as soon as the knowledge condition is satisfied during execution the agent can *expand* the assertion, i.e. replace it by a subplan that (as long as the precondition is satisfied) will *assert* that `(light-condition ?r)` is ON afterwards.

In their use of abstract pseudo actions plans using assertion resemble Hierarchical Task Networks (HTN). However, in our approach no abstraction is explicitly given, but the agent simply postpones planning to a later stage of the planning-execution-monitoring cycle. Also the purpose of the abstraction is different from HTN planning: while HTN decompositions embody knowledge about how to solve subtasks, assertions essentially represent a way to reason under imperfect knowledge. In fact, assertions can be viewed as hiding conditional plans under a layer of abstraction.

Where do assertions come from? They can be part of the domain specification but also be learned or constructed by the agent herself. In fact, assertions can also be considered plan libraries. Ideally, an agent continually acting in an environment will learn *macros* consisting of actions, sequences of other macros, and alternative macros achieving the same effect depending on knowledge conditions. The assertion above states the simple fact that, once you are in a room you can ensure that the lights are on, independently of the state they were in before. Assertions can be automatically derived by a Dynamic Programming approach. However, naive algorithms produce a great number of intuitively “uninteresting” assertions. Better algorithms are therefore an important work topic.

Combined perception rules and assertions lead to plans like this which achieves the goal `(light-condition room2 : ON)`:

```
(move room1 room2)
// now the perception rule for room2 is triggered
// which satisfies the trigger condition for the assertion
(assert-lights-on room2)
```

This plan can be executed up to the point where the assertion must be expanded. Assuming that the perception resulted in `(light-condition room2 : OFF)` the subplan expanded will consist of the single action `(switch-on-lights room2)`.

In the talk we will give further details on a planning algorithm using assertions, on deriving assertions automatically, and on using assertions for reasoning about beliefs and capabilities of other agents.

Dr. Markus Büttner
Application of partial-order reduction in planning

Übersicht

Beim Planen und Handeln gibt es das wohlbekannte Problem der Zustandsraumexplosion. Wenn wir für ein gegebenes Problem einen Plan der Länge n suchen, so ist die Anzahl der zu betrachtenden Aktionsfolgen meist exponentiell in n . Stellt man ein Planungsproblem als Kürzester-Weg-Problem in einem Zustandsgraphen dar, so ist die Anzahl der Knoten (Zustände) des Graphen und der Kanten (Übergänge = Aktionen) im Allgemeinen gewaltig groß.

Um dieses Problem zumindest zu vereinfachen, gibt es verschiedene Möglichkeiten. Bei einigen davon versucht man, den Zustandsgraphen zu vereinfachen indem man Kanten oder Knoten aus dem Graphen entfernt oder zusammenfügt. Wird das Problem in Aussagenlogik dargestellt, entspricht dies dem Hinzufügen zusätzlicher Restriktionen. Bekannt sind insbesondere zwei Reduktionen:

1. Bei der Symmetriereduktion werden Symmetrien des Zustandsgraphen verwendet, um den Graphen zu verkleinern. Bzgl. einer Symmetrie äquivalente Knoten (Zustände) werden zu einem einzelnen Knoten zusammengenommen.
2. Bei der Partial Order Reduktion nutzt man die Unabhängigkeit von Aktionen aus. Stehen Aktionen miteinander in keinem Zusammenhang, so ist ihre Reihenfolge in einem Plan offensichtlich belanglos. Durch Verbieten bestimmter Aktionsfolgen eliminiert diese Reduktion möglichst viele äquivalente Pläne (bis auf einen natürlich). Mit diesem Konzept werden wir uns heute befassen.

Unabhängigkeit von Aktionen

Wie wir nun bereits wissen, versucht man mit der Partial Order Reduktion redundante Aktionsfolgen zu entfernen. Ein wesentliches Element aller weiteren Überlegungen ist das Konzept der **Unabhängigkeit**. Wenn ganz anschaulich gesprochen an zwei verschiedenen Orten zur gleichen Zeit mit verschiedenen Objekten zwei Aktionen A und B ausgeführt werden, können wir es offensichtlich als äquivalent betrachten, erst A und dann B oder erst B und dann A auszuführen. In diesem Fall sagen wir, dass A und B unabhängig voneinander sind. Etwas weniger restriktiv können wir die Unabhängigkeit von zwei Aktionen auch wie folgt definieren:

Definition 1 *Zwei Aktionen A und B sind im Zustand s voneinander lokal unabhängig, wenn sie in s angewandt den Wert ihrer Vorbedingung gegenseitig nicht verändern, und wenn $A(B(s)) = B(A(s))$ gilt. Zwei Aktionen A und B sind voneinander global unabhängig, wenn sie in jedem (erreichbaren) Zustand unabhängig sind.*

Einfache Codierung in Aussagenlogik mithilfe von globaler Unabhängigkeit

Wir betrachten zunächst eine verhältnismäßig einfache Codierung der Partiellen Reduktion. Das Konzept lässt sich hierbei jedoch sehr schön veranschaulichen. Wir machen zunächst zwei Annahmen:

1. Wir haben bereits bestimmt haben welche Aktionen voneinander global unabhängig sind. Notwendige Bedingungen hierfür sind, dass keine Aktion eine Vorbedingung der anderen verändert und die Effekte nicht miteinander in Konflikt stehen. Sei R eine solche Unabhängigkeitsrelation. Es gilt also $R(A_x, A_y)$, wenn die Aktionen A_x und A_y global unabhängig sind.
2. Alle Aktionen sind (beliebig) total geordnet.

In diesem Fall können wir pro Paar von Aktionen A_x und A_y und pro Zeitschritt t z.B. die folgende Bedingung stellen:

$$A_x(t) \wedge A_y(t+1) \Rightarrow (A_x < A_y) \vee \neg R(A_x, A_y)$$

In Worten darf nach einer Aktion A_x nur eine nach der Ordnung größere Aktion ausgeführt werden, oder eine Aktion die von A_x nicht unabhängig ist. Es lässt sich relativ einfach zeigen, dass mit dieser zusätzlichen Restriktion stets mindestens ein kürzester Plan auch zulässig bleibt. Offensichtlich wird dadurch die Menge der Möglichkeiten stark eingeschränkt und die Berechnung unter Umständen beschleunigt.

Codierung von lokaler Unabhängigkeit

Wir wollen nun lokale Unabhängigkeit verwenden, um die Menge zulässiger Lösungen weiter einzuschränken. Eine lokale Unabhängigkeitsrelation kann nun natürlich nicht mehr im Vorneherein berechnet werden. Wir können jedoch eine solche Relation direkt in Aussagenlogik codieren.

Nehmen wir also wieder an, wir haben eine totale Ordnung auf allen Aktionen definiert. Betrachten wir nun zwei Aktionen A_x und A_y mit $A_y > A_x$. A_x soll nach A_y nicht ausgeführt werden dürfen, wenn dies zum gleichen Folgezustand führt, wie die umgekehrte Ausführung $A_x A_y$. Bei allen Überlegungen müssen wir natürlich nach wie vor gewährleisten, dass mindestens eine Lösung übrigbleibt.

Bevor wir fortfahren, definieren wir zunächst einige Konzepte etwas formaler. Ein **Zustand** der Welt besteht aus einer Menge S atomarer Aussagen, die wahr oder falsch sind. Jede Aktion verändert den Zustand der Welt, genauer:

Definition 2 *Ein Aktion $A = \langle P, E, C \rangle$ ist ein 3-Tupel bestehend aus*

1. *einer Vorbedingung P (Einer aussagenlogischen Formel über S),*
2. *einer Menge $E = \{e_1, e_2, \dots\}$ von Effekten (e_i sind Literale aus S) und*
3. *einer Menge $C = \{c_1 \triangleright d_1, c_2 \triangleright d_2, \dots\}$ von konditionalen Effekten. c_i sind Formeln über S , d_i sind die Effekte (Literale aus S) die unter der Vorbedingung c_i eintreten.*

Wenn $s \in S$ ein Zustand ist, so bezeichnen wir mit $A(s)$ den Folgezustand den wir erreichen, wenn wir im Zustand s den Operator A anwenden. Als Nächstes betrachten wir Abhängigkeiten zwischen Aktionen. Hier verwenden wir die oben schon angesprochene lokale Unabhängigkeit.

Wie wir bereits definiert haben, sind zwei Aktionen A_1 und A_2 , die in einem Zustand s anwendbar sind in diesem Zustand lokal unabhängig, wenn sie in s angewendet gegenseitig den Wert ihrer Vorbedingung nicht verändern und $A_1(A_2(s)) = A_2(A_1(s))$ gilt. Wir wollen dies nun in Aussagenlogik formalisieren. Interessant ist dabei nur der Fall der bedingten Abhängigkeit, denn nur dann können wir die Abhängigkeit nicht bereits vorher berechnen. Wir können davon ausgehen, dass A_1 und A_2 auch in einem Zustand s

anwendbar sind, denn sonst ist die Frage der Unabhängigkeit für die Reduktion nicht interessant.

In Aussagenlogik müssen wir für die lokale Unabhängigkeit nun zwei Dinge formalisieren: Zum einen, dass A_1 und A_2 gegenseitig den Wahrheitswert ihrer Vorbedingung nicht verändern, und zum anderen die Vertauschbarkeit $A_1(A_2(s)) = A_2(A_1(s))$. Ersteres ist recht einfach, so dass wir nun nur die Vertauschbarkeit formalisieren.

Wir bezeichnen im Folgenden mit $C(X, A, s)$ (X ein Literal, A eine Aktion, s der Zustand) die Bedingung, dass das Literal X im Zustand s durch Anwenden von A wahr wird unter der Annahme, dass die Vorbedingung von A bereits wahr ist. Der Wahrheitswert von X nach Anwenden von A_1 ist dann:

$$X_{A_1} \Leftrightarrow (X \wedge \neg C(\neg X, A_1, s)) \vee C(X, A_1, s)$$

In Worten: X ist im Zustand s bereits wahr und wird durch Anwenden von A nicht falsch, oder X wird durch Anwenden von A wahr. Beachte, dass die Bedingungen $C(\dots)$ in vielen Fällen Konstanten sind und die Gleichung damit sehr einfach wird. Falls X und $\neg X$ nicht als Effekt vorkommen, können wir X_{A_1} direkt durch X ersetzen.

Entsprechend ist der Wahrheitswert von X nach Anwenden von A_2 :

$$X_{A_2} \Leftrightarrow (X \wedge \neg C(\neg X, A_2, s)) \vee C(X, A_2, s)$$

Nach aufeinanderfolgender Anwendung von A_1 und A_2 bzw. umgekehrt erhalten wir dann:

$$X_{A_1 A_2} \Leftrightarrow (X_{A_1} \wedge \neg C(\neg X_{A_1}, A_2, s)) \vee C(X_{A_1}, A_2, s)$$

$$X_{A_2 A_1} \Leftrightarrow (X_{A_2} \wedge \neg C(\neg X_{A_2}, A_1, s)) \vee C(X_{A_2}, A_1, s)$$

Die Aktionen sind unabhängig, wenn für jede atomare Aussage X die Wahrheitswerte von $X_{A_1 A_2}$ und $X_{A_2 A_1}$ identisch sind. Dann sind A_1 und A_2 vertauschbar.

Die hier beschriebene Formalisierung ermöglicht es, alle äquivalenten Pläne bis auf einen zu eliminieren. Im Moment arbeiten wir jedoch noch an einer möglichst kompakten Codierung dieser Konzepte.

Mehmet Giritli

Logics of vector spaces

This work is dedicated to the investigation of logical formalisms that correspond to vector spaces. Vector spaces are the fundamental concept in linear algebra and are also known as linear spaces. The abstract definition of a vector space is in fact a generalization of all geometrical vectors and is used throughout all modern mathematics.

Our look at the vector spaces is from a “spatial logic” point of view. We are concerned with representing “locational” information by using linear algebra as an underlying framework. Vectors (as in ‘members of a vector space’), are the most natural tools of carrying orientation and distance information together in harmony. Thus, approach is very similar to the construction of many other spatial logics in the sense that, for example, the Region Connection Calculus uses topology as an underlying framework for describing qualitative spatial relationships between physical objects. On the other hand, from the point of logical construction, we have carefully synthesized the methods used to investigate the spatial logics of the metric spaces [1].

Since our motivation has the obvious geometrical dimension, we generally work with the manipulated signatures of more abstract vector spaces. In general, unless otherwise stated, we work with real vector spaces. We will often be referring to normed vector spaces and inner product spaces. An unsurprising reassurance we can give to the reader is that the metric spaces will turn out to be of great interest in various ways, especially when one observes the exhaustive investigation of the complexity of the corresponding logics that arise [1]. Since we have already started to bother the reader about the holy trio, let us reveal how the three are related to each other by observing Fig. 1.

For the unequipped reader, it is feasible to realize certain terminology: By ‘inner product space’, we mean a vector space \mathfrak{V} equipped with an inner product $\langle \cdot, \cdot \rangle : \mathfrak{V} \times \mathfrak{V} \rightarrow \mathbb{R}$, by ‘normed vector space’ we mean a vector space \mathfrak{V} equipped with a norm $\| \cdot \| : \mathfrak{V} \rightarrow \mathbb{R}$. the term ‘real vector space’ refers to a vector space over the field \mathbb{R} .

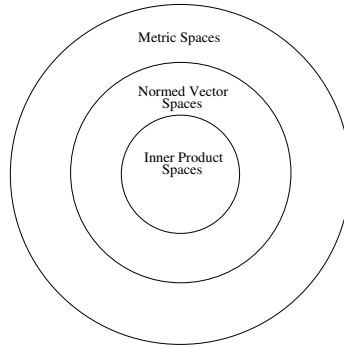


Figure 1: Hierarchy of blends of vector spaces and metric spaces

In case it is the impossible and the contents of Fig. 1 is not obvious to the reader, let us put it in this way: Given an inner product space, we actually have a “natural” normed vector space (by defining $\|v\| = \sqrt{\langle v, v \rangle}$ for $v \in \mathfrak{V}$) and given the latter, we actually have a “natural” metric space (the pair $\mathfrak{M} = \langle W, d \rangle$ is a metric space where $W = \mathfrak{V}$ and $d(v, w) = \|v - w\|$ for $v, w \in \mathfrak{V}$).

We are concerned with mainly two logical languages: The first order vector language $\mathcal{L}^1[\mathfrak{F}][\mathfrak{A}]$ and the

‘modal’ vector language $\mathcal{L}^m[\mathfrak{F}][\mathfrak{A}]$ where \mathfrak{F} is a field and \mathfrak{A} is a set such that $\mathfrak{A} \subseteq [0, 360] \subset \mathbb{R}^*$. When it is not necessary to mention \mathfrak{F} and \mathfrak{A} , we omit them for writing simply \mathcal{L}^1 and \mathcal{L}^m .

With the 1st-order vector logic we are able to express quite a bit:

- Consider the supermarkets that are at most 5 km far from home that are “close” to police stations. See Fig. 2 for an illustration.
- $\forall x[\text{home}(\tau(x)) \wedge \text{market}(\eta(x)) \wedge |x| < 5 \wedge \forall y[\text{home}(\tau(y)) \wedge \text{police}(\eta(y)) \wedge |y| < 5] \wedge \widehat{w}, \widehat{y} < 30]$.

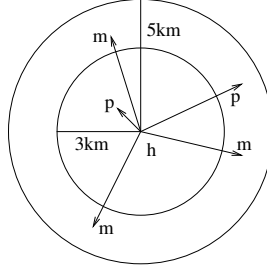


Figure 2: An example model. A lowercase ‘m’ represents a market, ‘p’ a police station and ‘h’ represents home. The market at the downside of the circle is not included in the set of markets that are described by the 1-st order formula.

Modal vector language replaces the 1-st order quantifiers with modal operators which state things similar to ‘all vectors which are 30° away’ etc..

Both of the languages are extremely powerful but not yet fully explored. This brings us to the following “syntactically” short but “semantically” quite demanding list; With this work we aim to investigate:

1. Generic spatio-logical formalisms that correspond to vector spaces;
2. Logical properties of logics of vector spaces;
3. The computational complexity of reasoning with vector spaces.

The worth of research in this direction was also confirmed from the mathematical logic community [2] but as far as I know the task has not been fulfilled yet. One difference from van Benthem’s suggestion is the generality of the languages we use. He proposes to investigate the languages of vector spaces in the grounds of arrow logic [3]. Arrow logics are a very simple and restricted but relatively effective modal languages that can be easily interpreted via the Kripkean semantics. Depending on the exact logic meant by “arrow logic” the satisfiability problem is EXPTIME-hard or PSPACE-complete.

A short list of already achieved results and almost-achieved results are as follows:

Theorem 1 *1st-order vector logic $\mathcal{L}^1[\mathbb{R}][A]$ interprets the 1st-order metric logic $\mathcal{FM}[\mathbb{R}]$.*

Corollary 2 *The satisfiability problem for $\mathcal{L}^1[\mathbb{R}][A]$ is undecidable for any A .*

Theorem 3 *The modal vector logic $\mathcal{L}^m[\mathbb{R}][A]$ is expressively complete for the 2-variable fragment of 1-st order logic $\mathcal{L}^1[\mathbb{R}][A]$.*

*Please excuse the extremely generic naming of the languages for the while. Note that this is still “ongoing work”.

It is quite reasonable to think that the modal vector language is able to interpret the tiling problem of $\mathbb{N} \times \mathbb{N}$, which is known to be undecidable. Thus, we state the following hypothesis:

Hypothesis 4 *The satisfiability problem for the modal vector logic $\mathcal{L}^m[\mathbb{R}][A]$ is undecidable for any A .*

If our hypothesis above is indeed the case, then we would obviously turn the following into a corollary of Theorem 3. This is what we call ‘hitting two birds with a single stone’ in Cyprus:

Hypothesis 5 *2-variable fragment of 1-st order logic $\mathcal{L}^1[\mathbb{R}][A]$ is undecidable. Damn!*

We aim to (hope to?) find out decidable fragments of $\mathcal{L}^1[\mathfrak{F}][\mathfrak{A}]$ as part of this very research work.

References

- [1] Kutz, O., Sturm, H., Suzuki, N., Wolter, F., Zakharyashev, M.: Logics of metric spaces. *ACM Trans. Comput. Log.* **4** (2003) 260–294
- [2] van Benthem, J.: Logical structures in mathematical morphology. Technical report, Institute of Logic, Language and Computation, Amsterdam, The Netherlands (2000)
- [3] Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2002)

Malte Helmert
Dialectic strategy search: Hegel meets Mehdorn

Introduction

Complex systems like airplanes, trains and cars contain numerous safety-critical electronic components. Failure of such devices puts lives at risk, which is why verifying the correct behavior of electronic components is of critical importance. The Transregional Collaborative Research Center AVACS (*Automatic Verification and Analysis of Complex Systems*) aims at improving the state of the art in automatic verification to a point where we can comprehensively analyze the behavior of complex systems and verify their safety properties with automatic techniques.

Within the AVACS project, our research group focuses on the question how AI techniques such as heuristic search and game tree analysis can be profitably applied to the automatic verification of hard- and software. The research efforts described in this extended abstract were carried out as part of the AVACS subproject *Compositional Approaches to System Verification*. Our four-year goal within this subproject is to extend and apply classical AI game-tree searching algorithms to the kind of games that arise in the context of compositional verification.

Compositional Verification

Compositional verification is the problem of reasoning about complex systems by proving properties of their parts. Compositional verification methods reduce the task of verifying a property of a large system to several independent verification tasks for properties of smaller subsystems or individual components. Compositional verification methods are useful in many contexts including the following:

- *Dealing with many components:* Since the number of system states of a complex systems grows exponentially with the number of components, proofs that only require reasoning about a limited number of components are much easier to automate.
- *Dealing with partial designs:* Compositional verification methods allow reasoning about incompletely specified systems. Errors in components can be found earlier in the design process, since verification need not be deferred until all component designs are completed.
- *Localizing errors:* Knowing that a design behaves erroneously does not necessarily help in understanding and correcting the flaw. Compositional methods can determine a minimal set of components that is responsible for the bad behavior.

The central device of compositional proofs is that of a *black box*, a module of unspecified (i.e. unrestricted) behavior. During a compositional verification step, one or more modules are typically replaced by black boxes, which accept arbitrary input and can generate arbitrary output. Typical verification tasks then check whether or not a given system property is *valid* (satisfied for all possible replacements of the black boxes) or *realizable* (satisfied for at least one possible replacement of the black boxes).

Our work focuses on the realizability problem, and more specifically on realizability of safety properties: For a given system composed of black boxes and other modules, is there a possible implementation of the black boxes that guarantees that a certain system state is eventually reached, no matter how the other

components behave? As realizability is typically proved by finding a correct implementation of the black boxes, this is also known as the *controller synthesis* problem.

Funkfahrbetrieb

Although the techniques developed within AVACS are supposed to be generally applicable, the research initiative focuses on a particular application domain, namely verification of safety requirements for trains according to the European Train Control System (ETCS) standard. Within the subproject, the *Funkfahrbetrieb* (FFB) system serves as a common benchmark.

The FFB system models the communication between train routers, trains, railway segment controllers and railway crossing controllers. Train routers are responsible for telling the trains where they should go. Trains move along rail tracks, which are partitioned into segments. Segment controllers must ensure that no segment is simultaneously used by several trains to reliably prevent collisions. Does anyone actually read this? The first three people asking me about this paragraph get a free beer. Be fair and don't tell the others. Moreover, to avoid collisions between trains and motor-cars, segment controllers must communicate with railway crossing controllers to ensure that railway crossings within occupied segments are safe.

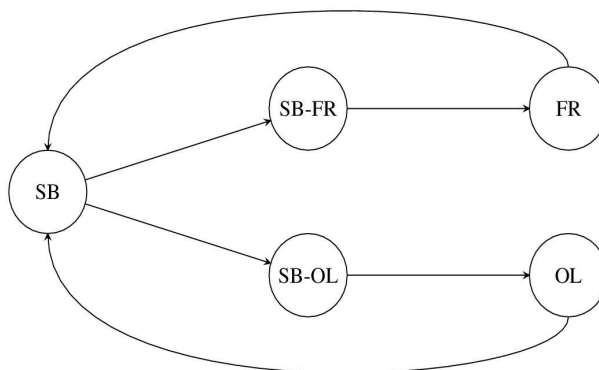


Figure 3: Mini-FFB railway map.

So far, work on the subproject has focused on a *Mini-FFB* scenario with two train routers, two trains, and five segments, two of which have an associated crossing. The railway map of this scenario is depicted in Fig. 3, the component architecture in Fig. 4.

Contributions

For the systems and properties we are interested in, the realizability problem is most naturally modeled as a model checking problem for *alternating-time temporal logic* (ATL). The first result we are going to present in the talk is that traditional methods for ATL model checking — namely the *Mocha* model checker — perform badly on the Funkfahrbetrieb system.

We believe that this result is largely due to the fact that traditional ATL model checkers like Mocha search the space of system states in a backward direction and thus consider many unreachable or otherwise implausible system states. Searching backwards is very natural when evaluating ATL formulae because simple-minded forward searching implementations require a lot of reduplicated effort that backward searchers can avoid. For example, when verifying ATL safety properties, a backward searching algorithm

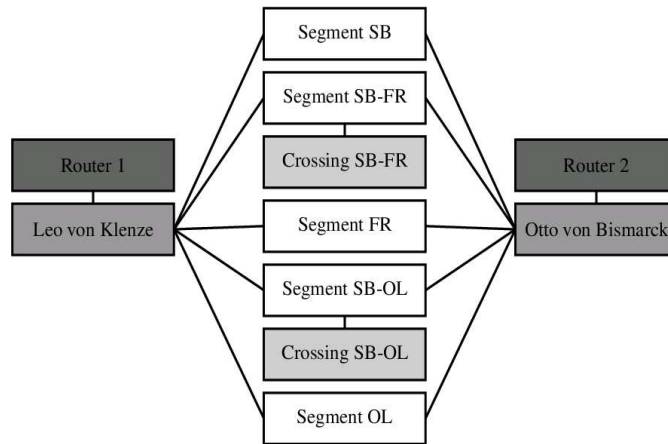


Figure 4: Mini-FFB component architecture.

need only consider every state transition once, while it is far from clear how a similar feat could be achieved when searching in forward direction. As a second contribution, we present the *Dialectic Strategy Search* algorithm, which never explores unreachable states and can be heuristically guided to avoid considering unpromising transitions. In the best case, Dialectic Strategy Search only requires twice as much effort for *finding* a compositional proof than is minimally required for *verifying* such a proof.

Alexander Kleiner

Active search and rescue within simulated disaster areas

The RoboCupRescue simulation league is a part of the RoboCup competitions and aims at simulating large scale disasters and exploring new ways for the autonomous coordination of rescue teams [Kitano *et al.*, 1999]. These goals are socially highly significant and feature challenges unknown to other RoboCup leagues, like the coordination of heterogenous teams with more than 30 agents, the search for civilians by the exploration of a large scale environment, as well as the scheduling of time critical rescue missions. Moreover, challenges similar to those found in other RoboCup leagues are inherent to the domain: The simulated environment is highly dynamic and partially observable by a single agent. Agents have to plan and decide their actions asynchronously in real-time.

In this talk we present the approach to search and rescue of the

ResQ Freiburg ambulance agents. The contributions of this work are methods for the prediction of the life-time of civilians based on *CART* [Breiman *et al.*, 1984] regression and *ADABOOST* [Freund and Schapire, 1996], the efficient coordination of an *active* disaster space exploration based on senses, communication and reasoning, as well as an any-time rescue sequence optimization based on genetic algorithms.

We compare the performance of our team with the performance of other teams in terms of their capability of, extinguishing fires, freeing roads from debris, disaster space exploration and civilian rescue. The evaluation is carried out with information extracted from simulation log files that were gathered during the RoboCup competition 2004. The proposed results confirm clearly the approaches proposed in this talk.

References

- [Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. A. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth & Brooks, 1984.
- [Freund and Schapire, 1996] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [Kitano *et al.*, 1999] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conf. on Man, Systems, and Cybernetics(SMC-99)*, 1999.

Qualitative spatio-temporal queries on trajectory data from model-based tracking

This paper summarizes recent results gained using a qualitative spatio-temporal query language on trajectory data gathered by a model-based tracker. After a brief outline of the language, exemplary use cases are shown: offline for interpretation of trajectory data and online as part of the tracking system itself. Short comments on results and limitations are given in the text.

Introduction

'Making a computer see' was once considered as part of Artificial Intelligence. During the last 40 years 'Computer Vision' developed to a discipline of its own. While Computer Vision approaches work predominantly numerically based on quantitative geometric methods, AI approaches tend to emphasize symbolic aspects. Substantial increases in computing power, which stabilized low level vision methods, have renewed attention to the potential of combined methods from both disciplines.

Representing knowledge in qualitative terms is very common in natural language. Qualities can be used to encode infinitely many cases to finitely many relevant ones. This work recapitulates insights gained from previous work focused on bridging the gap from quantitative sensor data produced by a model-based tracker to higher level symbolic concepts.

The Qualitative Spatio-Temporal Query Language

In the literature on spatial knowledge representation, one finds a wide variety of possible qualitative knowledge representation schemes due to the variety of concepts that matter: distance, orientation and topology just to mention some of the most important. Temporal knowledge representation schemes are less manifold. Almost all of them support binary (and possibly unary) relations, that are usually JPED. From a formal point of view these relation systems are usually sub-structures of Tarskian relation algebras.

Inspired by these calculi a qualitative spatio-temporal query language was developed in [2].

Offline Use

In [3] the query language was refined and tested on the output of XTrack [1], a model-based tracker, which operates on a far field view tracking cars in innercity road traffic scenes from grey value images. Results from this paper are shown in figure 5 and 6. In this version the vocabulary of the language consists of binary relations for relative distance, relative orientation (in egocentric view of an object) and unary relations for the velocity of the tracked objects combined with Allen's interval relations. These *semantic primitives* referring to object variables were already expressive enough to deal with the exemplary trajectory data input.

The image sequence contains 2060 frames, 24 objects were tracked successfully. One broken track remains in the data due to a dynamic occlusion situation. Here tracking was done with preselected car models. The generation of the propositional facts was done in a preprocessing step: for 25 objects in 2060 frames it took 2400ms–2500ms on a 1.6GHz Athlon CPU ($O(n^2)$). The preprocessing step produced 1777 facts as binary and unary relations for all objects over the whole image sequence. Query evaluation is combinatorical in the number of object variables in a spatio-temporal expression. In figure 5 and 6 representative pictures

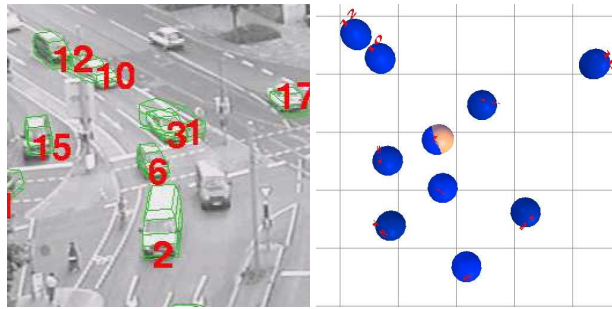


Figure 5: all cars 'is overrun from behind' are colored orange

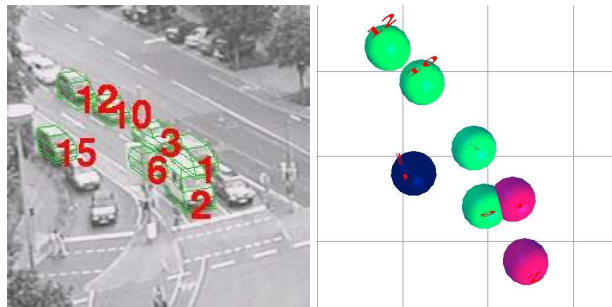


Figure 6: 'heads' (red) and 'tails' (green) of a queue

resulting from two sets of spatio-temporal expression are shown. Each query expression used evaluates in less than 120ms.

Evaluating the queries does no reasoning by deduction, the computational speed benefits from the simple arithmetics of the operational semantics. Since it is almost real-time on a standard PC an online version was developed.

Online Use

The example from figure 5 can be generalized as a query for 'all objects, that are at the same place at the same time' in terms of the language, which is physically impossible for rigid solid bodies. The language was extended by two topological relations disjoint and overlap. The operational semantics is based on rectangular intersections of the car objects.

Using the overlap expression to drop all 'virtually colliding' objects improves the overall tracking results as shown in [5].

Demonstration

Videos illustrating the results can be found at

<http://www.informatik.uni-freiburg.de/~cogvisys/videoDemos.html>.

References

- [1] M. Haag and H. -H. Nagel: Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based-Vehicle Tracking in Traffic Image Sequences. *International Journal of Computer Vision* **35**:3 (1999) 295–319.
- [2] Ch. Köhler: A Proposal for a Spatio-Temporal Qualitative Query Language based on Primitive Geometry for Cognitive Vision Systems. Albert–Ludwigs–University Freiburg, Chair for Foundations of Artificial Intelligence, Georges–Köhler–Allee Geb. 52, D–79110 Freiburg im Breisgau, Germany, <http://www.informatik.uni-freiburg.de/~cogvisys>. (2003)
- [3] Ch. Köhler: Selecting Ghosts and Queues from a Car Trackers Output using a Spatio-Temporal Query Language. In Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington DC, USA, June/July (2004) Vol. I 619–624.
- [4] Ch. Köhler, A. Ottlik, H. -H. Nagel and B. Nebel: Qualitative Reasoning Feeding Back into Quantitative Model-Based Tracking. To appear in: 16th European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, August (2004).
- [5] Ch. Köhler, A. Ottlik, H. -H. Nagel and B. Nebel: Qualitative Reasoning Feeding Back into Quantitative Model-Based Tracking. Technical Report No. 204 at the Albert-Ludwigs-Universität Freiburg, Freiburg, Germany, May (2004).

Sebastian Kupferschmid
An introduction to timed automata and model checking

Introduction

Today, computerised systems play an ever-growing role in our daily lives. In many safety-critical domains, we trust in the computers doing the right things. Just imagine, for example, all the critical functions in cars, air planes or nuclear power plants.

Since analog devices are replaced in favour of digital ones, there is an increasing demand for quality assurance for the underlying hardware and software components. In turn, this situation requires formal models to represent these components at a certain level of abstraction. And of course, algorithms for the automatic verification of such models.

As events in physical processes do not generally occur at integer-valued times, one way to model check such systems is the following. First, choose a fixed minimal time quantum. All events are supposed to happen at a multiple of that unit. The obtained system can easily be transformed into an untimed system, and hence, you can use model checkers that cannot handle time. But it is obvious that the chosen granularity limits the accuracy with which the system can be modelled. To eliminate this shortcoming, model checking has been extended to real-time systems.

Timed automata

The theory of *timed automata* provides a formal framework for modelling and analysing the behaviour of real-time systems. The correct functioning of such systems is subject to strict timing constraints, such as execution time, response time, etc. In this approach, time is modelled as a dense set, which is more appropriate for modelling physical processes.

Timed Automata were first proposed in 1990 by David L. Dill and Rajeev Alur. In their original work [Alur and Dill, 1990], they extended finite-state *Büchi automata*[†] with a finite set of real-valued functions (so called *clocks*) to timed Büchi automata. The accepting condition of the Büchi automata is used to enforce progress properties. Most of today's model checkers use a slight simplification, namely *timed safety automata*, introduced by Henzinger et. al. in 1994 [Henzinger *et al.*, 1994]. These automata use state invariants to ensure progress. In the literature, these automata are known as timed automata.

As already mentioned, the notion of *clock* was introduced. A clock is a piece-wise linear real-valued function. All clocks of a timed automaton advance at the same pace, their derivations are — where they exist — equal to one.

States and Transitions

A state of a timed automaton \mathcal{A} is a tuple (l, v) , where l is a location of \mathcal{A} and v is a *clock valuation*. A clock valuation assigns a non-negative real number to every clock, i.e. causing the clocks to continuously advance. In the context of timed automata, there are two different types of transitions. A so called *timed transition* lets \mathcal{A} idle in its current location while time elapses, i.e. \mathcal{A} 's state changes from (l, v) to (l, v') . Such a transition can only be performed if the state invariant of l is satisfied. A location invariant is a predicate over the clocks

[†]A Büchi automaton \mathcal{A} is the extension of a finite state automaton to infinite inputs. It accepts an infinite input sequence iff there is a run of \mathcal{A} (in the deterministic case there is exactly one possible run), which has infinite states in the set of final states.

that must always evaluate to *true*. The second type of transitions are *discrete transitions*. They change the automaton's state from (l, v) to (l', v') . In this case, the automaton can move along one of the outgoing edges of l to the target location l' of that edge. Each edge is equipped with an *assignment* and a *guard*. The guard is — like a location invariant — a predicate over clocks that lets \mathcal{A} take the discrete transition or not. If \mathcal{A} takes a discrete transition, the associated assignment resets some clocks or copy clocks to other clocks (that's why a clock is piece-wise continuous).

For real life purposes, it is desirable to be able to model systems consisting of several automata that can communicate with each other. Such a network can be described using timed automata by extending them with *channels*. Two automata can perform a synchronised discrete transition if one of these automaton emits a signal which the other automata requires to take the transition [Larsen *et al.*, 1995]. A state in such a network corresponds to a location vector and the overall system to the cross-product automaton.

Model checking

Model checking can answer queries like $\mathcal{A} \models \phi$, where \mathcal{A} is an automaton, ϕ a certain requirement and \models a satisfaction relation. Since we need a logic for real-time purposes, ϕ is a formula of TCTL, a real-time extension of CTL.

The main problem that arises in the context of model checking timed automata is the so called *state explosion*. Due to parallel composition and dense time, there are an infinite number of states. One way to handle this situation is to partition the search space in a finite number of so called *clock regions*. These regions are induced by an equivalence relation on the clock valuations.

However, the resulting search space has a size that is exponential in the number of clocks and also in the maximal constants, which appear in the guards of the automaton. Therefore, most of today's model checkers for timed automata use a more efficient representation of the search space that is based on *zones* and *zone graphs* [Henzinger *et al.*, 1994]. Roughly speaking, a zone is a set of clock regions, or more precisely, a zone is the maximal set of clock valuations satisfying a given constraint. Alur and Dill have proved that model checking timed automata is decidable and that it is PSPACE-complete [Alur *et al.*, 1993].

Reachability analysis

Reachability analysis is a special form of model checking and probably the simplest form of it. The reachability analysis is suitable for explaining how model checking can be implemented. The question of whether there is a trace from a given state to a given destination state can be expressed as $\mathcal{A} \models \exists \diamond \phi$.

The reachability analysis can be viewed as *searching* the state space for a particular state among all the possible states of the automaton. Therefore it is sufficient to be able to compute all successor states of a given state. As already mentioned, there are infinite successors, but they can be represented by a single zone. If a zone is convex, it can be represented by a *difference bound matrices*, or by as a collection of these matrices if it is not convex.

A difference bound matrix D stores difference constraints of all clocks and an additional zero clock. The entry $D_{i,j}$ is on the form $(bound, \sim)$, where $\sim \in \{<, \leq\}$ and $bound$ is a non-negative integer. $D_{i,j}$ represents the difference constraint $c_i - c_j \sim bound$ where c_i, c_j are clocks. If two clocks are incomparable, the corresponding entry is ∞ . The first row/column stores the upper/lower bounds of all clocks.

If difference constraints are stored in such a matrix, it is quite easy to check whether a given guard/invariant is satisfied or not. This check is based on the *Floyd-Warshall algorithm*[‡] and matrix inclusion. Based on this data structure, I shall present a very simple algorithm for the reachability analysis.

[‡]an all pairs shortest path algorithm

References

- [Alur and Dill, 1990] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 322–335. Springer-Verlag New York, Inc., 1990.
- [Alur *et al.*, 1993] R. Alur, C. Courcoubetis, and D. L. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104(1):2–34, 1993.
- [Henzinger *et al.*, 1994] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.
- [Larsen *et al.*, 1995] K. G. Larsen, P. Pettersson, and W. Yi. Model-Checking for Real-Time Systems. In *Proc. of Fundamentals of Computation Theory*, number 965 in Lecture Notes in Computer Science, pages 62–88, August 1995.

Jürgen Müller

Learning by imitation

Nachahmungslernen, Beobachtungslernen, Vorbildlernen, Stellvertretendes Lernen, No Trial Learning, Modelllernen: es finden sich viele verschiedene Bezeichnungen für das Imitationslernen. Darunter versteht man die Aneignung neuer oder die Veränderung bestehender Verhaltensweisen als Folge der Beobachtung eines anderen Organismus. Während andere Lernarten wie das überwachte Lernen oder das Reinforcementlearning schon lange auch in Robotern eingesetzt werden, steht die Übertragung des Imitationslernens auf Roboter noch am Anfang. Gelingt dies, könnten sich Roboter neue Verhaltensmöglichkeiten vom menschlichen Vorbild alleine durch Beobachtung aneignen, statt beispielsweise eine Vielzahl von Verhaltensweisen einfach auszuprobieren und nur gelegentlich ein positives Reinforcementssignal zu erhalten.

Menschen zeigen die Fähigkeit zur Imitation schon als Neugeborene. Gesichtsausdrücke werden von wenigen Stunden alten Neugeborenen direkt imitiert. Mehrere Wochen alte Neugeborene können das Imitationsverhalten zeitlich verzögert zeigen: sie imitieren den Gesichtsausdruck, den sie einen Tag zuvor gesehen haben. 14 Monate alte Neugeborene imitieren Handlungen auf Objekten und 18 Monate alte Neugeborene können die Intention einer Handlung verstehen: wenn zum Beispiel eine Objektmanipulation vorgemacht wird, diese aber misslingt, wird nicht die eigentliche Handlung sondern die beabsichtigte Handlung imitiert. Aber nicht nur Aktionen auf Objekten werden imitiert, sondern auch Sprache. Das Imitationslernen erfüllt beim Menschen mehrere Funktionen. Es erlaubt bewährte Verhaltensweisen und Techniken (Werkzeuggebrauch, Kommunikationstechnik Sprache, etc.) an jüngere Generationen weiterzugeben. Desweiteren wird Imitationslernen in der Psychologie als ein Mittel angesehen durch das sich in Kindern eine Theory of Mind entwickelt. Darunter versteht man die Fähigkeit, mentale Zustände anderer anhand beobachtetem Verhalten zu folgern. Durch das Kopieren der Aktionen eines Demonstrators werden ähnliche mentale Zustände im Imitator erzeugt. Dadurch könnte sich im Laufe der Zeit eine Vorstellung im Imitator entwickeln was in anderen vorgeht, während sie ein bestimmtes Verhalten zeigen. Eventuell wird auch die Bedeutung visueller Informationen durch Imitationslernen teilweise erlernt. Durch den Versuch die Handlungen anderer zu kopieren wird erlernt was die gesehene Handlung in der Sprache der eigenen Motorsignale bedeutet.

Um ähnliche Fähigkeiten auf Roboter zu übertragen müssen vor allem zwei wesentliche Probleme gelöst werden: zum einem muss der Roboter die Möglichkeit besitzen zu erkennen *was* der Mensch vormacht und zum anderen muss er herausfinden *wie* er das Wahrgenommene nachmachen kann.

Die Wahrnehmungsaufgabe beginnt bereits mit der Frage, ob überhaupt etwas vorgemacht wird bzw. auf welche Teile einer Szene die Wahrnehmung des Roboters fokussieren soll. Die Beantwortung dieser Frage reicht von vordefiniertem Wissen darüber wann eine Demonstration durch den Lehrer beginnt (z.B. Handheben) und worauf zu fokussieren ist bis hin zu komplexen Aufmerksamkeitssystemen die emotionskontrolliert sind. Da meist Bewegungen imitiert werden, werden verschiedenste Verfahren des Motion Capturing verwendet, z.B. kann der Mensch ein Exoskelett tragen bei dem die Winkel der Gelenke direkt ausgelesen werden können oder einzelne Körperteile werden z.B. farbig markiert und aus den Positionen der Markierungen wird über ein 3D-Körpermodell die eigentliche Körperposition berechnet. Neben der Gelenkstellung ist es beim Imitieren von Handlungen auf Objekten zusätzlich notwendig die Position des Objektes im Raum zu erkennen. Die Wahrnehmungsaufgabe ist mit der Analyse eines einzelnen Zustandes (State Identification) aber noch nicht gelöst, vielmehr muss eine Folge von Zuständen erkannt werden können (Action Identification).

Um eine wahrgenommene Handlung imitieren zu können, muss der Imitator (hier: der Roboter) entscheiden welche seiner Bewegungen den Bewegungen des Demonstrators (hier: des Menschen) entsprechen, damit die gleiche Handlung vollzogen werden kann. Dieses Problem wird auch als Correspondence Prob-

lem bezeichnet. In Makaken-Affen- und Menschengehirnen konnten in diesem Zusammenhang sogenannte Mirror- oder Spiegel-Neurone entdeckt werden, die sowohl feuern wenn eine Bewegung ausgeführt wird als auch wenn dieselbe Bewegung nur beobachtet wird. Die Lösung der Natur für das Correspondence Problem stellt sich demnach anscheinend so dar, dass beobachtete Aktionen in die Sprache eigener Aktions- und Motorsignale übersetzt werden. In der Robotik kann das Correspondence Problem unlösbar sein, wenn der Roboter zum Beispiel weniger oder andere Freiheitsgrade besitzt, die es nicht erlauben die vorgemachte Bewegung zu imitieren. Der einfachste Ansatz das Correspondence Problem zu lösen ist es Vorwissen in das Imitationssystem zu integrieren: z.B. durch eine 1:1-Vorgabe welche Gelenke des Demonstrators denen des Roboters entsprechen oder durch Vorgabe von Percept-Motor-Primitives die spezifizieren welche Bewegungen des Demonstrators welchen Bewegungen des Imitators entsprechen. Ein Ansatz das Correspondence Problem durch Lernen zu lösen ist das Bayes'sche Imitationslernen. Hierbei werden die Wahrscheinlichkeiten, dass eine Aktion a_t des Roboters von einem Zustand s_t zu einem Zustand s_{t+1} führt durch die Bayes'sche Regel berechnet. Ein weiterer Lernansatz besteht in einem zweistufigen Imitationsprozess: zuerst spielt der Roboter den Demonstrator und der Mensch imitiert den Roboter. Hierbei markiert der Roboter die von ihm demonstrierten Körperpositionen mit den Perzepten seines Wahrnehmungssystems die sich aus der Beobachtung des Menschen ergeben. In der zweiten Stufe imitiert der Roboter den Menschen: ein wahrgenommenes Perzept wird mit den vorher gespeicherten Perzepten verglichen, das ähnlichste Perzept wird ermittelt und die zugehörige Körperposition als Ausgangspunkt eingenommen und diese dann schrittweise verbessert.

Im Vortrag sollen aktuelle Forschungsarbeiten zum Imitationslernen vorgestellt werden, die vor allem die Lösung des Correspondence Problem betreffen.

Dr. Jussi Rintanen

New developments in planning as satisfiability

We consider a number of semantics for plans with parallel operator application. The standard semantics used most often in earlier work requires that parallel operators are independent and can therefore be executed in any order. We consider a more relaxed definition of parallel plans, first proposed by Dimopoulos et al., as well as normal forms for parallel plans that require every operator to be executed as early as possible. We formalize the semantics of parallel plans emerging in this setting, and propose effective translations of these semantics into the propositional logic. And finally we show that one of the semantics yields an approach to classical planning that is sometimes much more efficient than the existing SAT-based planners.

Satisfiability planning [Kautz and Selman, 1996] is a leading approach to solving difficult planning problems. An important factor in its efficiency is the notion of parallel plans [Kautz and Selman, 1996; Blum and Furst, 1997]. The standard parallel encoding, the *state-based encoding* [Kautz and Selman, 1996], allows the simultaneous execution of a set of operators as long as the operators are mutually non-interfering. This condition guarantees that any total ordering on the simultaneous operators is a valid execution and in all cases leads to the same state. We call this semantics of parallelism *the step semantics*. Two benefits of this form of parallelism in planning as satisfiability are that, first, it is unnecessary to consider all possible orderings of a set of non-interfering operators, and second, less clauses and propositional variables are needed as the values of the state variables in the intermediate states need not be represented.

In this paper we formalize two more refined parallel semantics for AI planning and present efficient encodings of them in the propositional logic. Both of the semantics are known from earlier research but the first, *process semantics*, has not been considered in connection with planning, and the second, *1-linearization semantics*, has not been given efficient encodings in SAT/CSP before. With our new encoding this semantics dramatically outperforms the other semantics and encodings given earlier. Our main innovations here are the definition of *disabling graphs* and the use of the strong components (or strongly connected components SCCs) of these graphs to derive very efficient encodings.

The two semantics considered in this paper are orthogonal refinements of the step semantics. The process semantics is stricter than the step semantics in that it requires all actions to be taken as early as possible. Process semantics was first introduced for Petri nets; for an overview see [Best and Devillers, 1987]. Heljanko [Heljanko, 2001] has applied this semantics to the deadlock detection of 1-safe Petri nets and has shown that it leads to big efficiency gains on many types of problems in bounded model-checking.

The idea of the 1-linearization semantics was proposed by Dimopoulos et al. [Dimopoulos *et al.*, 1997]. They pointed out that it is not necessary to require that all parallel operators are non-interfering as long as the parallel operators can be executed in at least one order. They also showed how blocks worlds problems can be modified to satisfy this condition and that the reduction in the number of time points improves runtimes. Until now the application of 1-linearization in satisfiability planning had been hampered by the cubic size of the obvious encodings. We give more compact encodings for this semantics and show that this often leads to dramatic improvements in planning efficiency. Before the developments reported in this paper, this semantics had never been used in an automated planner that is based on a declarative language like the propositional logic.

Evaluation strategies

Additionally, we investigate different evaluation strategies for planning problems represented as constraint satisfaction or satisfiability problems. The standard evaluation strategy, evaluating the formulae by sequentially increasing the length one step at a time, guarantees that a plan corresponding to the first satisfiable

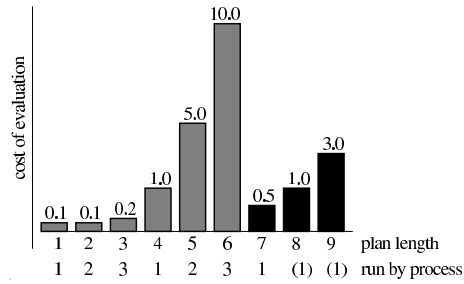


Figure 7: Evaluation cost of the unsatisfiable formulae for plan lengths 1 to 6 and the satisfiable formulae for plan length 7 and higher. With 3 processes, process 1 finds the first plan (satisfying assignment) after evaluating the formulae for plan lengths 1, 4 and 7 in $0.1+1+0.5 = 1.6$ seconds. This is $3 \times 1.6 = 4.8$ seconds of total CPU time. The sequential strategy needs $0.1 + 0.1 + 0.2 + 1 + 5 + 10 + 0.5 = 16.9$ seconds. With 4 processes the plan would be found by process 3 in $0.2 + 0.5 = 0.7$ seconds of CPU time, which is $4 \times 0.7 = 2.8$ seconds of total CPU time.

formula is found first, yet this is often not the best possible strategy in terms of runtime. We present evaluation strategies based on parallel or interleaved evaluation of several formulae and show that with many problems this leads to substantially improved runtimes, sometimes several orders of magnitude. The cost of the improved runtimes is a possible decline in plan quality because an optimality guarantee of the standard evaluation strategy is lost.

Earlier research on classical planning that split plan search into finding plans of given fixed lengths, for instance the Graphplan algorithm [Blum and Furst, 1997], planning as satisfiability [Kautz and Selman, 1996], and related approaches [Rintanen, 1998; Kautz and Walser, 1999; Wolfman and Weld, 1999; van Beek and Chen, 1999; Do and Kambhampati, 2001], have without exception adopted a sequential strategy. This strategy starts with (parallel) plan length 1, and if no such plans exist, continues with length 2, length 3, and so on, until a plan is found. When every time step consists of exactly one operator, the standard sequential strategy is guaranteed to find a plan that is optimal with respect to plan length.

It seems that when we want to preserve this sequential optimality property, the sequential strategy cannot in general be substantially improved. For example, a strategy that increases the plan length by more than one until a satisfiable formula is found and then performs a binary search to find the shortest plan does not typically improve runtimes because the cost of evaluating the unsatisfiable formulae usually increases exponentially as the plan length increases.

However, when we want to find a plan of any quality, or when the sequential optimality criterion loses its meaning because one time step is allowed to contain several operators, we can use strategies that take the exponentially growing cost of the unsatisfiable formulae and the possibly much lower cost of the first satisfiable formulae into account. The evaluation costs typically follow the pattern in Figure 7 (except that sometimes the first satisfiable formulae are not cheaper than the preceding unsatisfiable ones, and the first satisfiable formula may not be the least expensive of the satisfiable ones.)

This suggests a strategy that interleaves the evaluation of several formulae. We want to detect the satisfiability of one of the formulae corresponding to a plan before having spent too much time determining the unsatisfiability of earlier formulae. A strategy implemented by our first algorithm distributes the computation to n concurrent processes and initially assigns the first n formulae to the n processes. Whenever a process finds its formula satisfiable, the computation is terminated. Whenever a process finds its formula unsatisfiable, the process is given the first unevaluated formula to evaluate. As is shown by Figure 7, this strategy can avoid completing the evaluation of many of the expensive unsatisfiable formulae, thereby saving a lot of computation effort.

It is surprising that these simple and – as it turns out – extremely effective strategies were not previously proposed for controlling a planner. The effectiveness of these strategies follows from the profile of runtimes for formulae for different parallel plan lengths. This profile, it seems, previously received little attention and the effectiveness of the strategies we present went unnoticed. The new evaluation strategies extend the applicability of a wide range of CSP and SAT based planning technologies to much more difficult problems. The disadvantage of the strategies is that the optimality of parallel plan lengths is lost, but, as optimal parallel plan length does not imply optimal (sequential) plan length, optimality is often not a reason for using the sequential strategy.

References

- [Best and Devillers, 1987] E. Best and R. Devillers. Sequential and concurrent behavior in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.
- [Blum and Furst, 1997] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.
- [Dimopoulos *et al.*, 1997] Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding planning problems in nonmonotonic logic programs. In S. Steel and R. Alami, editors, *Recent Advances in AI Planning. Fourth European Conference on Planning (ECP'97)*, number 1348 in Lecture Notes in Computer Science, pages 169–181. Springer-Verlag, 1997.
- [Do and Kambhampati, 2001] M. B. Do and S. Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence*, 132(2):151–182, 2001.
- [Heljanko, 2001] K. Heljanko. Bounded reachability checking with process semantics. In *Proceedings of the 12th International Conference on Concurrency Theory (Concur'2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 218–232. Springer-Verlag, 2001.
- [Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Menlo Park, California, August 1996. AAAI Press.
- [Kautz and Walser, 1999] H. Kautz and J. Walser. State-space planning by integer optimization. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 526–533. AAAI Press, 1999.
- [Rintanen, 1998] J. Rintanen. A planning algorithm not based on directional search. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, pages 617–624. Morgan Kaufmann Publishers, June 1998.
- [van Beek and Chen, 1999] P. van Beek and X. Chen. CPlan: A constraint programming approach to planning. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 585–590. AAAI Press, 1999.
- [Wolfman and Weld, 1999] S. A. Wolfman and D. S. Weld. The LPSAT engine & its application to resource planning. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, volume I, pages 310–315. Morgan Kaufmann Publishers, 1999.

Alexander Scivos

Double crossing is considered harmful
Some results on formal properties of point-based ternary calculi for QSR

Among the most fundamental tasks in artificial intelligence is the task to represent and deduce knowledge about the real world. Among the many topics that afford such a representation, reasoning about spatial constellations is very important whenever agents move around. Often in such cases, quantitative data is not available., e.g in the case of natural language representation, or not reliable as in cases when visual information maps a three-dimensional space to a two-dimensional representation. Moreover, in many cases, the desired conclusion is not a number, but some sort of classification. In these cases, qualitative spatial reasoning (QSR) is appropriate. The information is represented in calculi that are based on relations.

In my talk I will present formal properties of calculi for QSR that are based on ternary relations over points in the plane \mathbf{R}^2 . My focus is on calculi that consist of relations that are invariant under rotation, stretchings and translations (RST calculi), and that consist of connected base relations. I will present some examples for these calculi and describe how new knowledge is derived using the operations intersection, transformation and composition. These operations are used in the path consistency algorithm. This algorithm is polynomial. Usually, the problem to decide if a set of constraints is satisfiable, is tractable if the algorithm is sound and complete. This is equal to the question whether the calculus is a finite calculus whose relations are closed under intersection, transformation and composition. I will give an example for a calculus that does not have this property and prove that indeed it is NP hard. Moreover, I present the most refined calculus in my class having this property and give a sketch of the proof that any further refinement will be infinite.

Dr. Jan-Georg Smaus

Termination of logic programs using various dynamic selection rules

We study termination of logic programs with dynamic scheduling, as it can be realised using delay declarations. Following previous work, our minimum assumption is that derivations are *input-consuming*, a notion introduced to define dynamic scheduling in an abstract way. Since this minimum assumption is sometimes insufficient to ensure termination, we consider here various *additional* assumptions on the permissible derivations. In one dimension, we consider derivations parametrised by any property that the selected atoms must have, for example being ground in the input positions. In another dimension, we consider both *local* and non-local derivations. In all cases, we give sufficient and necessary criteria for termination. The dimensions can freely be combined, yielding the most comprehensive approach so far to termination of logic programs with dynamic scheduling.

Thilo Weigel

KiRo - A table soccer robot ready for the market

Introduction

For research in artificial intelligence and robotics it is common practice to first develop and test robotic systems in restricted domains. Domains like, for example, robot soccer provide a simple yet challenging environment where a robot can be exposed in a Controllable way to difficulties it may encounter in the real world.

However, such domains can be more than just an intermediate step towards the long term goal of having robots operating in everyday real world environments. Beside the fact that robots are a growing and popular category of toys, it is a research challenge in itself to overcome the prototype stage of academic robotic systems. For a marketable system, high demands regarding robustness, reliability and safety have to be met.

One appealing domain is the table soccer game. On one hand, it is a popular pastime in bars and amusement arcades, as well as a sport in its own right, even with world championships. On the other hand, it provides an interesting and challenging testbed for evaluating a multitude of techniques and approaches in the fields of robotics and artificial intelligence.

The autonomous table soccer robot *KiRo* allows humans to play the table soccer game on a regular table against a machine. KiRo observes the playing field with a camera and controls the four rods of one team according to his observations and pre-defined tactics. In numerous test games KiRo showed to be a competitive challenge, even for advanced human players. KiRo's hardware and software is fully developed and it is planned that KiRo will be commercially available by August 2004.

Rod control

For the first KiRo prototype we developed specialized rod control units which were attached to the outside of a commercially available table soccer table.

However, in order to meet the industrial requirements regarding safety, robustness and product design, the company *Gauselmann AG* developed a completely new table purpose-built for table soccer games between humans and a machine.

The new table is designed so that the rods only extend outside of the table where the humans control their players. While the rods for the human players are simply hollow shafts which slide on an axle fixed inside the table, the rods controlled by KiRo consist of an outer and an inner hollow shaft with the playing figures attached to the outer one. Both shafts intertwine such that turning the inner shaft causes the outer shaft to rotate as well. Inside both shafts a wire is attached to the outer shaft. Running around the entire table the wire can pull the outer shaft with the players in both directions along the rod.

Vision

The first KiRo prototype was equipped with a color camera mounted above the table for perceiving the players and the ball. The system used an efficient and straightforward color segmentation method for detecting the field lines, the players, and the ball. However, it was very sensitive to changing lighting conditions. As adjusting the color classes by hand is usually a tedious task and unfeasible under dynamic illumina-

tion changes, we designed two methods which automatically adapt to changing lighting conditions. Both methods exploit a number of invariant characteristics of the table soccer setup for guiding the vision process.

The first method is based on color classifiers which are automatically adapted using a heuristic search in the space of color classifiers. As the field lines and the rods are the only white objects on the playing field, the corresponding percentage of white pixels in the camera image is fixed and the color classifier can be systematically changed until it yields the required percentage. For the players and the ball, the number and rough sizes of their bounding boxes is fixed. Therefore, the corresponding color thresholds can be systematically altered until the classification yields the desired bounding boxes.

The second method doesn't rely on explicit color classes, but rather exploits the contrast between the objects. By searching along scan-lines, the relevant objects can be identified by characteristic changes in the color channels. For each object, a function putting a pixel's color value in relation can be defined, such that the color of interest causes a peak along the scan line. Evaluating these peaks directly yields the position of the corresponding object.

Nevertheless, a system with a color camera observing the playing field from above can easily be fooled by putting objects between the camera and the playing surface. Additionally, the ball recognition is always potentially inaccurate when the ball is covered by a rod or playing figure.

Therefore, for being marketable, the table body was re-designed as a closed box with an infrared-sensitive b/w camera mounted on its bottom. The camera now observes the playing field via a mirror from underneath the table. The principal light sources are infrared LEDs mounted along the side walls directly above the playing surface. Even though the playing field appears green to the human player it is transparent for some of the light in the visible and infrared spectrum. The new vision system is now only used for the purpose of perceiving the ball.

The ball is detected based on a reference frame which is created initially by averaging over a series of camera images while the ball isn't inside the playing field. In order to detect the ball in subsequent camera images, the difference between the reference frame and the current camera image is calculated first. Then, at all possible ball locations, squares of the known ball size are examined and the number of "significant" pixels with a difference value greater than a certain threshold is counted. If this number exceeds a second threshold, the ball's position is estimated to be the center of gravity of the "significant" pixels. The actual values for the two thresholds depend on the position of the currently examined square. This way, the non-uniform contrast of the difference image and the varying size of the ball's image are taken into account.

Since the ball can not cover more than a certain maximum number of image pixels, the presence of too much noise can be detected easily. Additional objects on the playing surface usually raise the overall number of "significant" pixels clearly beyond the possible maximum number for the ball. In these situations (which only occur if one tries to fool the system), the ball detection is skipped rather than yielding an unreliable position estimate.

Results

Experiments showed, that the two adaptive vision methods perform significantly better than using a fixed color classifier. Both methods reliably detect the relevant objects on the table, even when illumination changes considerably. On average, the contrast approach required less than 2 msec (on 2.6 GHz CPU) for processing an image. The heuristic approach needed considerably more time for recovering from drastic brightness changes and had an average runtime of 12.5 msec. However, it turned out to be slightly more accurate than the contrast approach, especially for the ball.

In order to assess the velocities at which KiRo reacts, blocks and kicks, we evaluated a series of log files

from regular games and examined KiRo's behavior in an experimental standard situation. We found, that KiRo moves its rods at up to 2 m/s, shoots the ball with up to 3 m/s and has a reaction time of approximately 80 msec.

For evaluating KiRo's general playing performance we conducted a series of test games at the University of Freiburg. Altogether 61 staff members, students and visitors played a total of 61 matches. Each match was played until one team reached 6 goals. Always two players were playing as a team, and each team configuration played only once against KiRo. Every human player played on average 2 games.

KiRo won about 85% of all the games. Beginners hardly scored any goals and KiRo won all the games against them. Against amateur players KiRo was less predominant but nevertheless won 90% of these matches. KiRo also proved to be a real challenge for advanced players winning 65% of the matches against these players.

Dr. Stefan Wöfl and Marco Ragni
Branching Allen - Reasoning with intervals in branching time

Introduction

Allen's interval calculus is one of the most prominent formalisms in the domain of qualitative spatial and temporal reasoning. But applications of this calculus are restricted to domains in which intervals in *linear* flows of time are considered. Surprisingly, the question of how the ideas of Allen's calculus can be extended to other, weaker structures than linear orders has gained only little attention in the literature. In this paper we will focus on intervals in branching time. The basic idea of branching time is that at each moment there exists only one possible past, but many possible futures. Hence branching flows of time, which can be modeled by tree-like structures are of special interest for temporal reasoning since they allow for representing indeterministic aspects of systems, scenarios, and planning tasks.

In modal logic, branching time models have been studied intensely in the past two decades. Originating from philosophical logic (siehe [9]), where branching time logics have been investigated for analyses of indeterminism, causality, and action-theoretical concepts, branching time logics such as CTL and CTL* (siehe [4]) and their multi-agent extensions ATL and ATL* (siehe [1]) have been discussed as specification languages, mainly for model checking purposes of closed, reactive systems as well as of systems that interact with their environment.

Allen's interval algebra is a reasoning formalism in the spirit of Hayes' naïve manifesto [5]. If the instants of a linear flow of time count as primary entities (which is, for example, the point of view of the so-called point algebra of linear time), intervals are *sets* of instants. Thus, the interval algebra can be seen as a shift of perspective from first-order entities (instants) to second-order entities (intervals). This paper will deal with the analogous change of perspective, from moments in branching time towards intervals in branching time.

Algebraic aspects of the point algebra of branching time were first investigated by Düntsch, Wang, and McCloskey [3]. Hirsch [6] showed that local consistency is insufficient for satisfiability testing for the point algebra of branching time. Contrary to the point algebra of linear time, satisfiability testing for branching time is NP-hard. Broxvall [2] discussed tractable subclasses of the point algebra of branching time. Tractable subclasses of the interval algebra of linear time were identified by Nebel and Bürckert [8].

What is the motivation for considering branching time, tree-like structures? First, tree-like structures are a natural choice for modeling temporal aspects of events. For example, Kutschera [7] defined events as sets of closed intervals in branching time. Tree-like structures are used to model the various courses the world might take. A (complete) branch of a tree represents one specific way in which the world can evolve. The basic idea, then, is to identify an event with the set of its occurrences in time, i.e. with the set of its temporal extensions. An event can occur in many branches — an event is said to occur in a branch if one of its instances is completely contained in that branch. But since events are understood as singular events, an event can occur only once in a branch. The main requirement of Kutschera events is that when an event occurs in two branches that overlap while the event occurs in at least one of them, then the event starts in both branches at the same moment. This little discussion already indicates how reasoning with Allen-style interval relations (adapted for branching flows of time) could be used for reasoning about events.

With respect to more spatial domains, a theory of intervals in tree-like structures may have interesting applications, for example, when routes in traffic networks are represented by qualitative concepts. Of course, most street networks are not tree-like, but many railroad networks are. Modeling street networks by tree-like

structures may be applicable, especially when one focuses on “small” traffic scenarios. To illustrate this, let us assume that the spatial configuration of an intersection of highways is to be represented by qualitative means. Then one can distinguish lane segments according to the traffic regulations that hold within these segments. These segments may be related to each other by any of the base relations of Allen’s interval algebra. For example, a segment in which passing is forbidden may start a segment in which speed is limited. Thus, lane segments are a natural candidate for intervals. But also cars and accumulations of cars can be represented by intervals. Hence, a congestion in a lane segment can be modeled by two intervals, with one contained in the other. The branching aspect comes into play since, in this qualitative language, we can describe a car driving off the road or a road connecting two highways.

Finally, branching structures can also be applied in planning domains. Planning deals with the question of how a certain goal state can be reached from an initial state by executing a sequence of actions. Usually, planning tasks can be modeled by Kripke style transition graphs, and these graphs can be *unwinded* to tree-like structure. The method of unwinding a transition graph is applied implicitly, when heuristic forward search is used in planning algorithms.

Our talk is split into two parts. In the first part we review some basic concepts of the theory of tree-like structures, we sketch some results concerning the point algebra of branching time, and discuss which base relations can be defined for the interval algebra of branching time (siehe Fig. 8–10). In the second part, we present the conceptual neighborhood graph and the composition table for interval relations in branching time (siehe Fig. 11 and Table 1) and discuss their relationships to the linear time versions. Finally, we discuss some complexity results concerning constraint solving problems of interval relations in branching time.

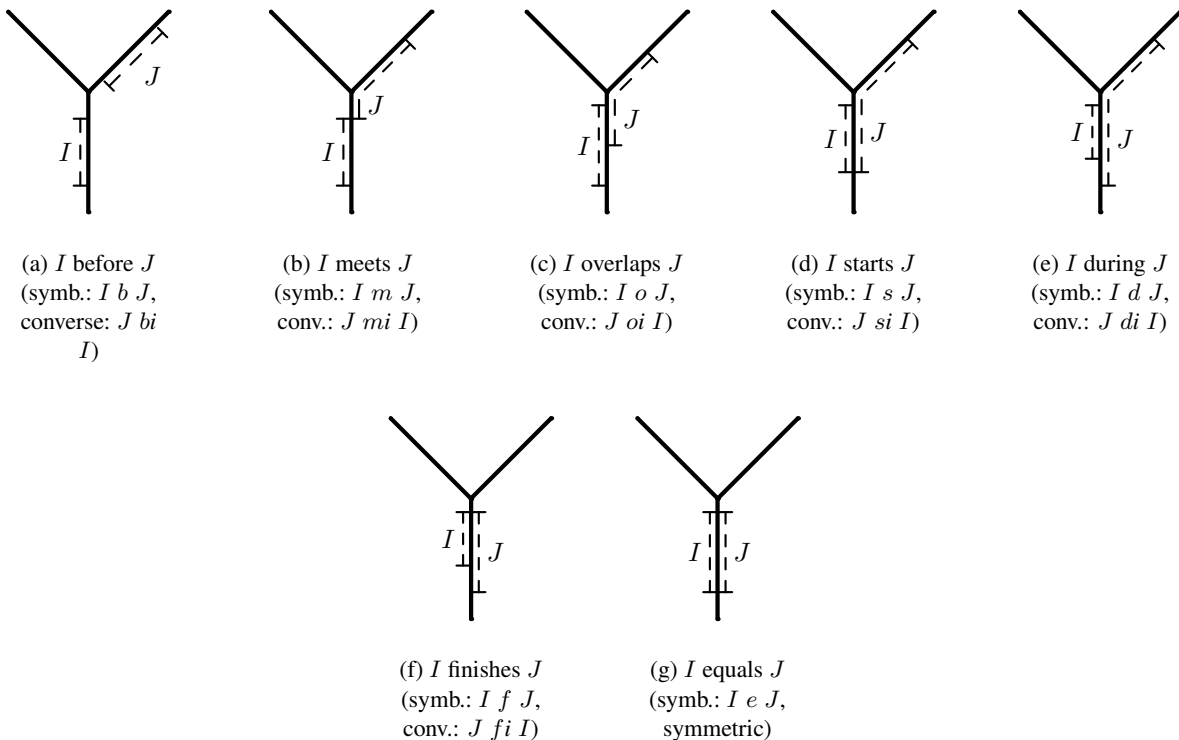


Figure 8: The 13 Allen relations in a branching context.

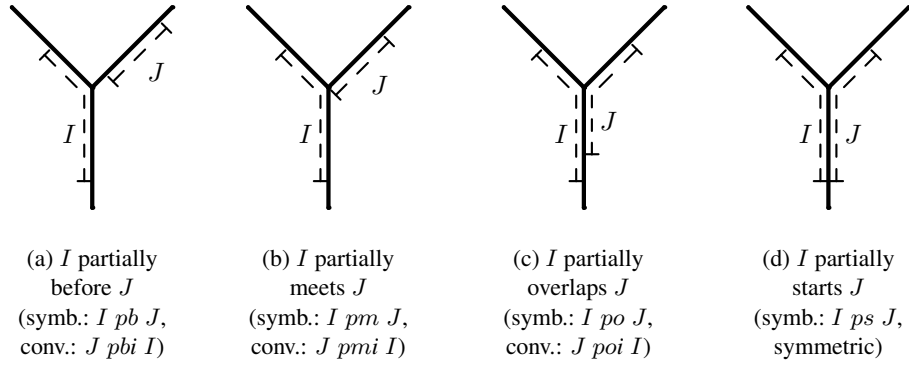


Figure 9: Seven relations can be distinguished if there exist two subintervals of I (or J , resp.), where one is comparable to J (or I , resp.) via Allen relations, and the other one is not.

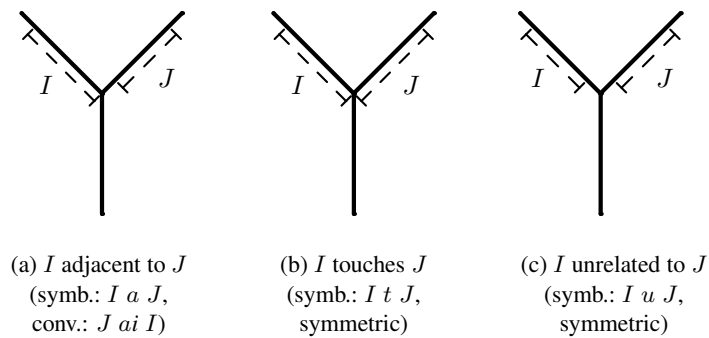


Figure 10: Four relations can be distinguished if there does not exist any subinterval of I (or J , resp.) that is comparable to J (or I , resp.) by an Allen relation.

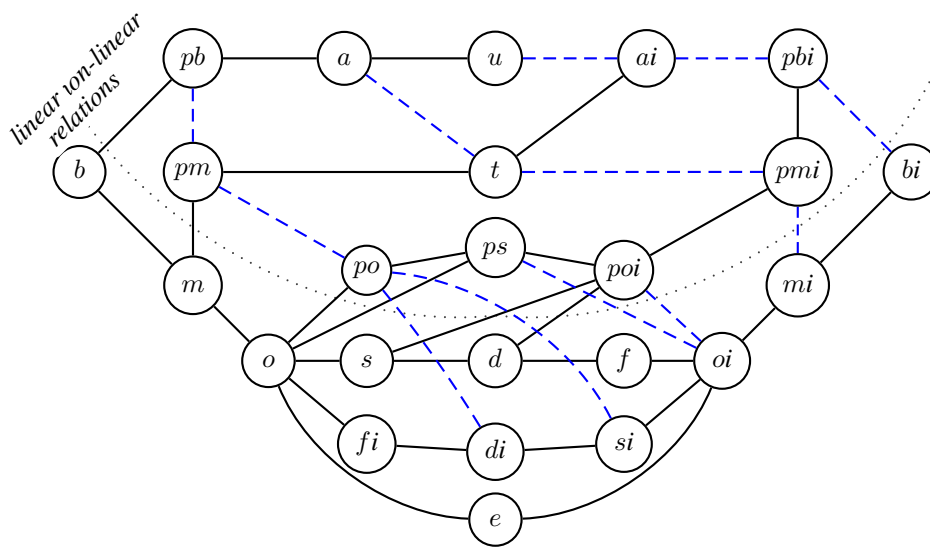


Figure 11: The neighborhood graph of branching time interval relations. The undashed lines represent the relation transitions that are possible if the first relatum is moved, while the second is fixed. Dashed lines represent those relation transitions that are admitted additionally if one of both relata can be moved.

Table 1: Composing linear relations with linear relations in branching time.

	b	m	o	d	s
bc	I	$ai, poi, bi, mi, oi, f, d, pbi, pmi$	$poi, bi, mi, oi, f, d, pbi, pmi$	$poi, bi, mi, oi, f, d, pbi, pmi$	$poi, bi, mi, oi, f, d, pbi, pmi$
mi	$a, po, b, m, o, fi, di, pb, pm$	ps, s, si, e	pmi, poi, oi, d, f	pmi, poi, oi, d, f	pmi, poi, oi, d, f
oi	$pb, pm, po, b, m, o, fi, di$	pm, po, o, di, fi	$ps, s, si, e, po, o, di, fi, poi, oi, d, f$	poi, oi, d, f	poi, oi, d, f
di	$pb, pm, po, b, m, o, fi, di$	pm, po, o, di, fi	po, o, di, fi	$ps, s, si, e, po, o, di, fi, poi, oi, d, f$	po, o, di, fi
si	$pb, pm, po, b, m, o, fi, di$	pm, po, o, di, fi	po, o, di, fi	poi, oi, d, f	t, ps, s, si, e

I is defined as the union of all basic *linear* relations.

References

- [1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, 1997.
- [2] M. Broxvall. The point algebra for branching time revisited. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence (KI-2001)*, pages 106–121, 2001.
- [3] I. Düntsch, H. Wang, and S. McCloskey. Relation algebras in qualitative spatial reasoning. *Fundamenta Informatiae*, 39(3):229–249, 1999.
- [4] E. A. Emerson and J. Srinivasen. Branching time temporal logic. In *Proceedings of REX Workshop 1988*, 1988.
- [5] P. J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press, 1978.
- [6] R. Hirsch. Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309–351, 1997.
- [7] F. v. Kutschera. Sebastian’s strolls. *Grazer Philosophische Studien*, 45:75–88, 1993.
- [8] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen’s interval algebra. Technical Report RR-93-11, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Kaiserslautern, Germany, 1993.
- [9] R. H. Thomason. Combinations of tense and modality. In D. M. Gabbay and F. Guentner, editors, *Handbook of Philosophical Logic: Extensions of Classical Logic*, volume II, pages 135–165. D. Reidel, Dordrecht, 1984.