Universität Bielefeld

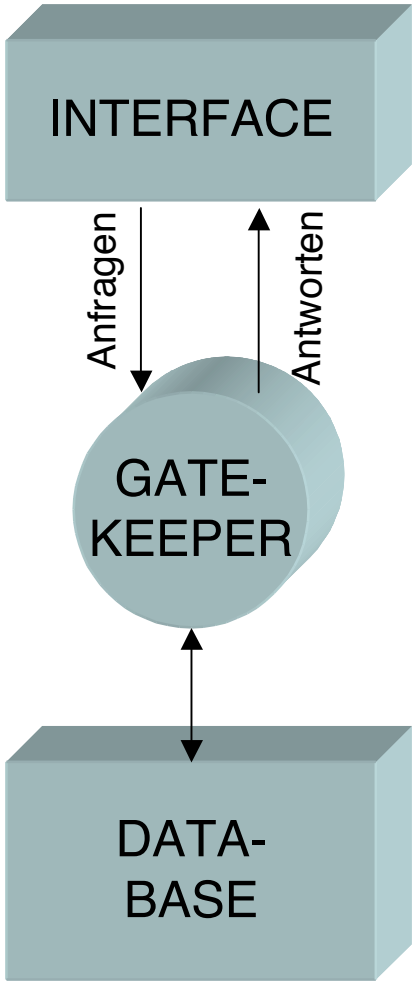
4 Spezielle Schlussverfahren

12. Vorlesung: Temporales und räumliches Schließen

Methoden der Künstlichen Intelligenz

Ipke Wachsmuth WS 2008/2009

Wozu temporales Schließen?



Zustände und Ereignisse

- **sind Begriffe, die implizit ein *Zeitkonzept* voraussetzen**
- **Zustände können sich ändern:**
  Es muss möglich sein, in der DATABASE buchzuführen darüber, was *zu bestimmten Zeiten* gilt.

Für Systeme, die aktiv auf eine äußere Welt einwirken können, um ein Ziel zu erfüllen:

- **Es soll möglich sein, hypothetische zukünftige Weltzustände (die mit bzw. ohne Eingriffe des Systems eintreten würden) berechnen zu können.**

--> *Projektionen*

Ansätze temporalen Schließens

Typische Anfragen über Zeitdaten:

- Ist ein Faktum während eines bestimmten Intervalls gültig?
- Ist ein Faktum nach einem Ereignis (noch) gültig?
- Wie ist die zeitliche Relation zwischen zwei Zuständen?

Ansätze zur Behandlung von Zeit:

- Resultatbezogene Darstellung: Situationskalkül
- Tokenbezogene Darstellungen:
  - intervallbasierte
  - punktbasierte

Ansätze: Overlap Chaining, TSA Zeitkartenverwaltung (TMM), Allens Kalkül

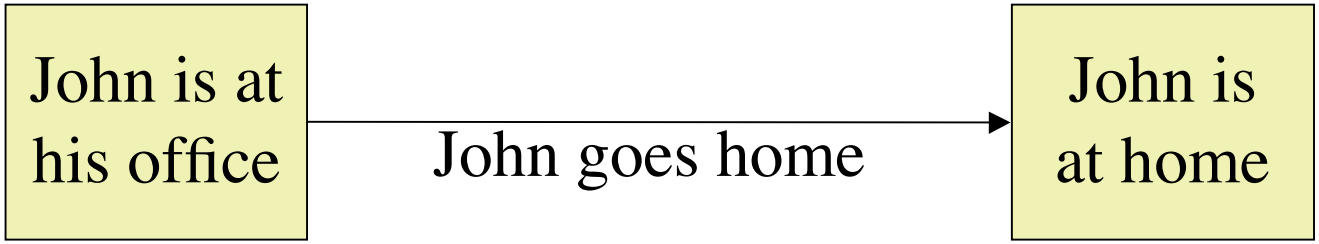
Leseempfehlung hierzu:

- Charniak & McDermott, Kapitel 7, Seite 416-433

in Ergänzung:

- Puppe, Einführung in Expertensysteme, Kap. 9

Der Situationskalkül



- Ereignisse als Übergänge (Transitionen) zwischen Zuständen in Situationen dargestellt
- Situation: Ein Zeitintervall, in dem kein (relevanter) Zustand seine Gültigkeit ändert
- neue Situation als Resultat des Eintretens eines Ereignisses

Zeitlogische Beschreibung

`(T s p):`	Zustand *p* ist wahr in Situation *s*.
`(not(T s p)):`	Zustand *p* ist falsch in Situation *s*.
`(result s e ):`	Situation, die sich bei Eintreten des Ereignisses *e* aus der Situation *s* ergibt.

Ausdruck für einen (kausalen) zeitlichen Zusammenhang:

Ereignis

```
(T (result ?s (do john (go (home john))))
   (loc john (home john)))
```

Zustand

Ausdruck für ein tatsächlich in einer Situation eintretendes Ereignis

```
(occur s-29 (do john (go (home john))))
```

Beispiel: Projektionen inferieren

eine „Eiswürfel-schmelz-Regel“

```
(if (and (inst ?x icecube)
         (T ?s (outside ?x))
         (T ?s (> (amb-temp) 0)))
    (occur ?s (melt ?x)))
```

eine „Nasswerde-Regel“

```
(if (and (inst ?x icecube)
         (T ?s (on ?x ?y)))
    (T (result ?s (melt ?x))
       (wet ?y)))
```

Liege in `s-21` ein Eiswürfel `cube-32` auf Tisch `table-41`

Inferenzen:

```
(occur s-21 (melt cube-32))
(T (result s-21 (melt cube-32))
   (wet table-41))
```

Schwierigkeiten bei Projektionen

Angenommen ein weiterer Eiswürfel `cube-33` wird in der Resultat-Situation des geschmolzenen `cube-32` eingeführt.

Kann geschlossen werden, dass auch er schmilzt?

NEIN, denn:

die eingeführten Axiome lassen den Schluss nicht zu, dass die Umgebungstemperatur `amb-temp` immer noch > 0 ist.

Es müsste eigentlich postuliert werden:

„Frame-Axiom“

```
(if (and (inst ?x icecube)
         (T ?s (> (amb-temp) 0)))
    (T (result ?s (melt ?x))
       (> (amb-temp) 0)))
```

Gelten von Rahmenbedingungen

„Frame-Problem“

(McCarthy & Hayes, 1969)

Das Frame-Problem betrifft die Spezifikation dessen, was sich <u>nicht</u> ändert, wenn ein Ereignis eintritt.

Alle Annahmen darüber, dass irgendwelche Zustände von den betrachteten Ereignissen nicht berührt werden, müssen in einer prädikatenlogischen Modellierung explizit ausgedrückt werden:

- **sog. *Frame-Axiome***
- **davon könnten sehr viele benötigt werden**

ACHTUNG – Unterscheide:

"frame" im Sinne von nicht berührten Rahmenbedingungen (hier)

"frame" als Bezeichnung für eine Wissensrepräsentation (früher)

Problem mit `result`-Notation

- **diskrete (als signifikant betrachtete) Zustände werden modelliert**

- **graduelle (stetige) Übergänge finden keinen Ausdruck**

Der Situationskalkül ist nur dann nützlich, wenn gerade EINE "nächste" Situation `(next` *`s`*`)` als Resultat eines in *`s`* aufgetretenen Ereignisses anzunehmen ist.

Wo solche einfachen Zusammenhänge gegeben sind, kann der Wechsel von jeweils einer "augenblicklichen" Situation*(current situation)* zur nächsten mit Hilfe von `addlist` und `deletelist` modelliert werden.

Ansatz mit `delete-/addlist`

`deletelist:` **Liste von Fakten, die bei Eintreten eines Ereignisses falsch werden (und in DATABASE gelöscht werden)**

`addlist:` **Liste von Fakten, die bei Eintreten eines Ereignisses wahr werden (und in DATABASE hinzugefügt werden)**

Probleme bei Daten-abhängig-keiten!

current situation:

```
(frozen cube-30)
(on cube-30 picnic-table-3)
(> (amb-temp) 0)
```

next situation:

```
(melted cube-30)
(> (amb-temp) 0)
(wet picnic-table-3)
```

```
deletelist:(frozen x)
addlist:   (melted x)
```

Token-bezogene Zeitdarstellung

zu verwenden, wo Situationen nicht als Resultat-orientierte Beschreibung von anderen Situationen und darin auftretenden Ereignissen repräsentiert werden können.

Allgemeines Prinzip:

- **Die Formeln in DATABASE müssen Zeitmarkierungen enthalten, die Aussagen zulassen, *wann* sie gelten.**
- **Fetch-Pattern müssen ebenfalls Zeitangaben enthalten**
- **Normalerweise sind die erfragten und zu findenden Zeit-Deskriptoren *nicht identisch***
- **Der Retrieval-Mechanismus muss also Zeitangaben vergleichen können.**

Ansatz des „Overlap Chaining“

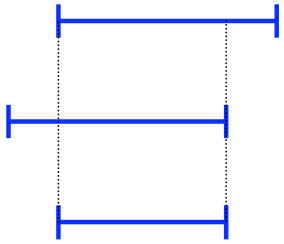
Time-Token **(eines Zustandes):**

ein bestimmter Zeitabschnitt, in dem der fragliche Zustand gültig (true) ist.

Spezielle Algorithmen und Datenstrukturen zur Beantwortung von zeitabhängigen Fragen sind erforderlich.

Einfaches Beispiel: overlap chaining

Wenn ***p*** **über die Zeitperiode** ***T1*** **gilt**
und `(if p q)` **über die Zeitperiode** ***T2*** **gilt,**
dann gilt ***q*** **über den Durchschnitt von** ***T1*** **und** ***T2,***
sofern ***T1*** **und** ***T2*** **überlappen.**



Zeitbehandlung: weitere Ansätze

- **Temporal System Analyzer (TSA)**

  Vorhersage (Zukunft) / Erklärung (Rückblick) von zu bestimmten Zeiten gültigen Fakten (betrifft GATEKEEPER)

(dazu gehört "Overlap Chaining"; weiteres bei Bedarf S. 422 – 429, Charniak/McDermott)

- **Time-Map Manager (TMM)**

  Verwaltung von zeitabhängigen Fakten (betrifft DATABASE)

(wird jetzt erläutert)

Zeitkarten (time maps): 2 Arten

- **intervallbasierte Zeitkarten**

  Knoten: Zeit-Tokens (mit einer Dauer)

  Kanten: Relationen zwischen Zeit-Tokens

  (überlappen, zusammentreffen, ...)

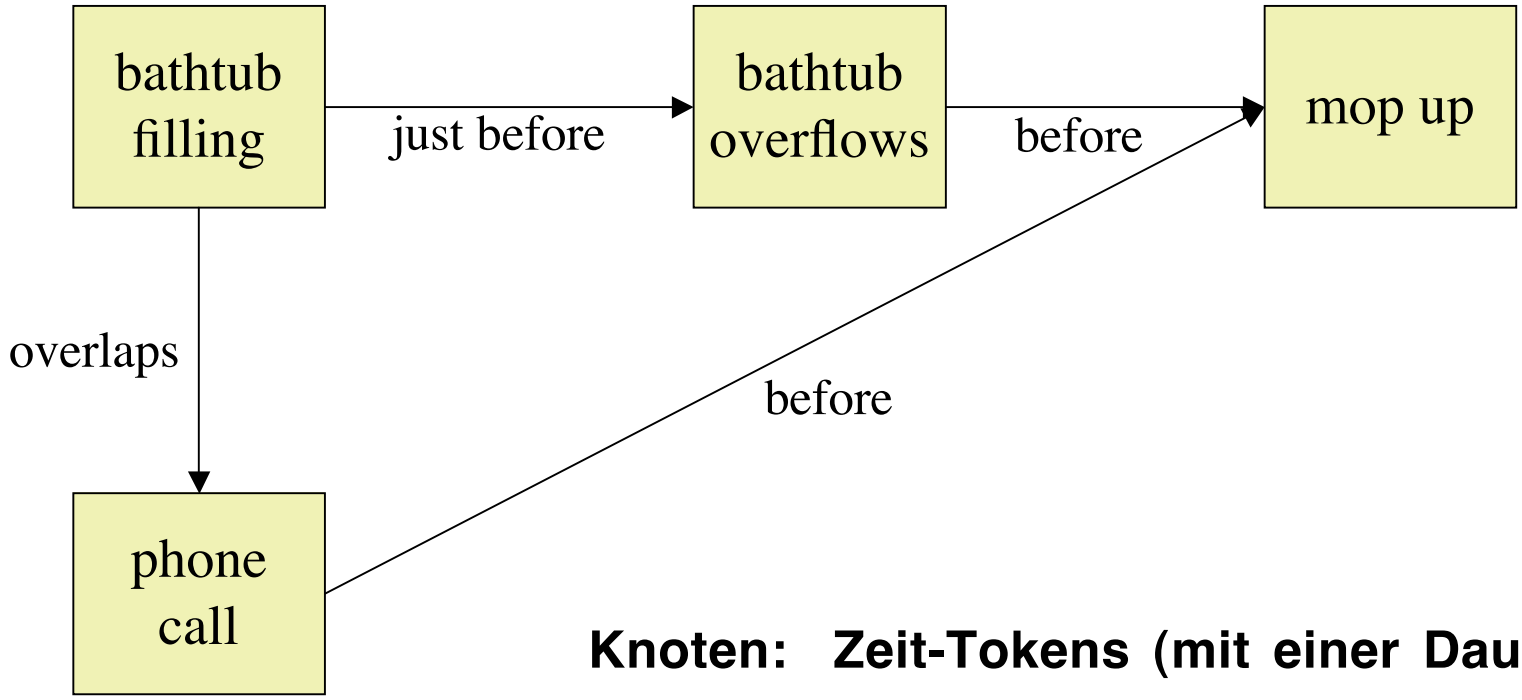
- **punktbasierte Zeitkarten**

  Knoten: für Situationen, die Anfang bzw. Ende von Zeit-Tokens sind

  Kanten: (nur) vorher und gleichzeitig

Die Zeitrelationen werden jeweils in einem assoziativen Netzwerk verwaltet.

Beispiel: Intervallbasierte Zeitkarte

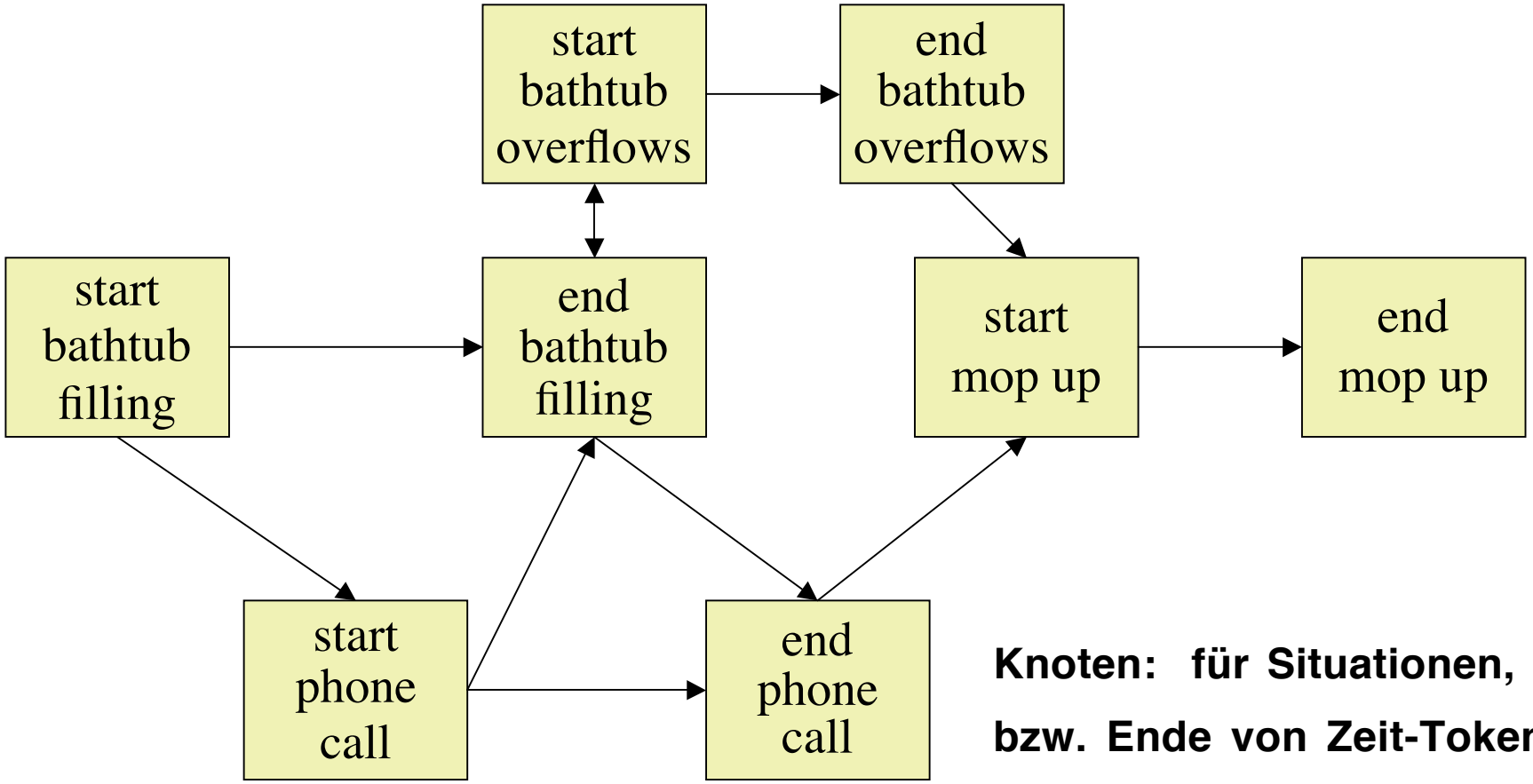


Knoten: Zeit-Tokens (mit einer Dauer)

Kanten: Relationen zwischen Zeit-Tokens

(überlappen, zusammentreffen, ...)

Beispiel: Punktbasierte Zeitkarte



Knoten: für Situationen, die Anfang bzw. Ende von Zeit-Tokens sind)

Kanten: vorher und gleichzeitig

Vorgehen und Probleme

Um Fragen über zu vergleichende Zeit-Tokens zu beantworten, sucht der TMM Pfade zwischen ihnen und vergleicht die an Kanten vermerkten Zeitrelationen.

(unterschiedliche Vor- und Nachteile bei punkt- bzw. intervallbasierten Karten)

- **Zuordnung von Zeitdauern oder Datumsangaben zu Intervallen oft sinnvoll.**
- **Bei großen Zeitkarten erheblicher Rechenaufwand für Graphensuche**

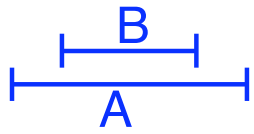
  Abhilfe:
  - **datierte Zeit-Tokens (unmittelbarer Vergleich)**
  - **hierarchisch organisierte Zeitkarten (grobe/feine Suche)**
- **Probleme ergeben sich vor allem bei *konjunktiven* zeitabhängigen goals.**

Nachteil punktbasierter Zeitkarten

> Unelegante Beschreibung qualitativer Zeitbeziehungen

Schreibweise `(gilt (distanz (Z1 Z2) 0 10)`
modelliert: „Zeitpunkt Z1 liegt zwischen 0 und 10 Zeiteinheiten vor Z2“

Ausgedrückt werde z.B. „Faktum B gilt zeitweilig während Faktum A“




```
(gilt (distanz (beginn A beginn B) pos_winzig pos_unendlich)
(gilt (distanz (ende B ende A)     pos_winzig pos_unendlich)
(gilt (distanz (beginn A ende A)   pos_winzig pos_unendlich)
(gilt (distanz (beginn B ende B)   pos_winzig pos_unendlich)
```

modelliert: „A beginnt vor B“ „B endet vor A“ „A ist ein Intervall“ „B ist ein Intervall“

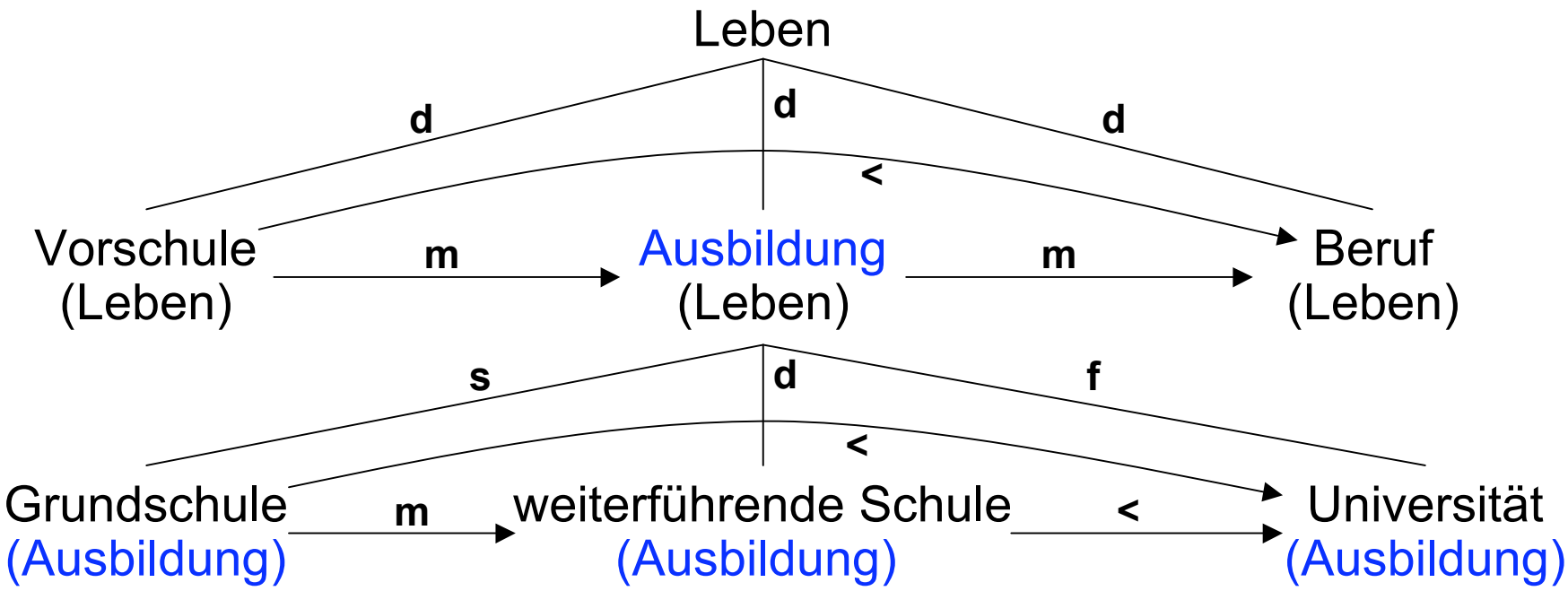
Besser beschreibbar in Allens Zeitkalkül: X *during* Y etc.

Allens Zeitkalkül: 13 Relationen

Relation	Symbol	inverse Relation	grafisches Bsp.
X *before* Y	<	>	
X *meets* Y	m	mi	
X *overlaps* Y	o	oi	
X *during* Y	d	di	
X *starts* Y	s	si	
X *finishes* Y	f	fi	
X *equal* Y	=	=	

- Intervallbasierte Zeitrepräsentation; qualitativ
- elegante Beschreibung vieler Zeitbeziehungen
- z.B. Aufbau hierarchisch geordneter Zeitkarten

Hierarchisch geordnete Zeitkarte



Beispiel: In welcher Zeitbeziehung stehen Vorschule und Grundschule?

- Vorschule *meets* Ausbildung
- Grundschule *starts* Ausbildung

- Vorschule *meets* Grundschule

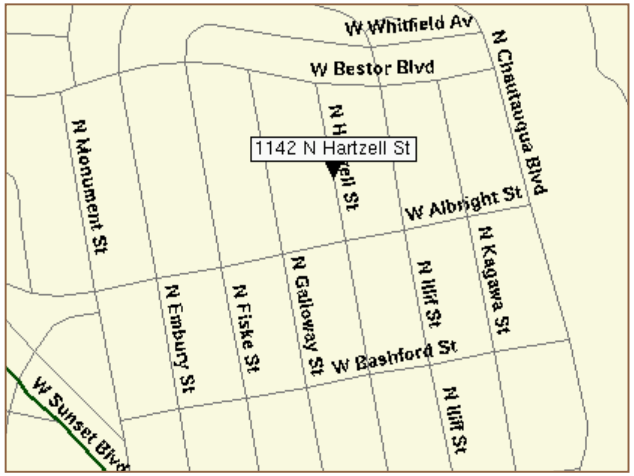
Räumliches Schließen: Einblick

- **Beispiel Wegsuche:**
  in Straßennetzen oder in einem mit Hindernissen besetzten Gelände von einem Punkt zu einem anderen gelangen

ist einfach insofern, als nur der veränderliche Ort eines Navigators in einer sonst statischen Umgebung zu betrachten ist; evtl. ist die Ausdehnung des Fahrzeugs beim Navigieren zu berücksichtigen.

<u>Zwei Basisansätze:</u>

- **pfadorientiert**
- **formorientiert**



Pfadorientierter Ansatz

führt auf hierarchische Graphsuche

- **Ebene der Städte**
- **Ebene der Stadtteile**
- **Ebene der Straßen**

Nur Minimalprogramm:

- Charniak & McDermott, Kapitel 7, Seite 433-437

Organisation in Graphen aus *Orten* und *Pfaden*

- **Jeder Ort (im Sinne von Stelle) wird einer Hierarchie-Ebene zugeordnet.**
- **Um z.B. den Ort "Rathaus" in A-Stadt von außerhalb aufzusuchen, wird zunächst auf der höchsten Ebene eine Verbindung (Pfad nach A-Stadt) gesucht und in A-Stadt auf Stadtteil- und schließlich auf Straßenebene verfeinert.**

–> hierarchischer route-finder-Algorithmus

Formorientierter Ansatz

Der pfadorientierte Ansatz funktioniert am besten, wo eine klare Unterteilung zwischen undurchquerbarem Gelände und passierbaren Verkehrswegen besteht.

Der formorientierte Ansatz ist angebracht, wo ein (nicht durch Straßen vorgezeichneter) Weg an Hindernissen (= den freien Durchgang behindernde Objekte) vorbei geplant werden muss.

Benötigt wird dann Information über

- **die Form von Objekten (soweit bekannt)**
- **ihre ungefähre Position und Orientierung**
- **Art der Verbindungen/Abgrenzungen von Objekten**

Die beiden Hauptfälle

1. **Wegsuche in einem überwiegend leeren Gelände**
   Aufgabe: Auffinden der repräsentierten *Objekte* in DATABASE; planen, wie herum oder zwischendurch zu manövrieren ist.

2. **Wegsuche in einem überwiegend vollen Gelände**
   Aufgabe: Auffinden der repräsentierten *Durchgänge* in DATABASE; planen, wie Durchgänge zu verfolgen sind.

Offenes Problem:

Wie lassen sich aus formorientierten Verfahren *Pfade* gewinnen, so dass - wo möglich - auf Graphsuche zurückgegriffen werden kann?