

Einführung in die Methoden der Künstlichen Intelligenz

Allens Zeitintervallogik

PD Dr. David Sabel

SoSe 2014

- Darstellung und Inferenzen für zeitliche Zusammenhänge
- Viele verschiedene Logiken
- z.B. Modallogiken und Temporallogiken
(stellen meistens genaue Zeitpunkte dar)

Wir betrachten die

Allensche Intervall-Logik

Beispiel: Käse-Sahne-Kuchen backen

Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



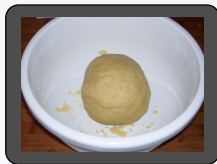
Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



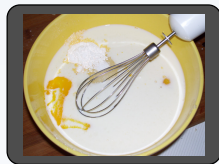
Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



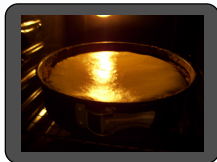
Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Beispiel: Käse-Sahne-Kuchen backen



Zeitliche Zusammenhänge



- Aktionen entsprechen (nicht-leeren) **Zeitintervallen**
- Wissen: Anforderungen an die **relative** Lage der Intervalle
- Wie kann man dieses Wissen **repräsentieren**?

Beispiele



vor



Zutaten bes.

Teig zub.



direkt nach



Teig zub.

Teig ruht



während,
aber vorher
endend



Sahne schl.

Belag zub.



beginnt
während



Belag zub.

Teig ruht

Welche Schlüsse lassen sich daraus ziehen?

- Neue Beziehungen zwischen Aktionen

Darf der Belag vor dem Teig in die Form?

- Modell: Anordnung der Intervalle, die alle Beziehungen erfüllt

Wie gelingt der Kuchen?

- Konsistenz: Gibt es ein Modell?

Kann man den Kuchen überhaupt backen?

Welche Schlüsse lassen sich daraus ziehen?

- Neue Beziehungen zwischen Aktionen

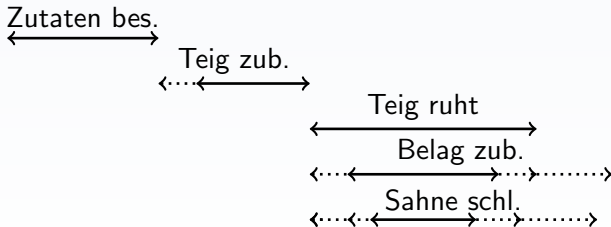
Darf der Belag vor dem Teig in die Form?

- Modell: Anordnung der Intervalle, die alle Beziehungen erfüllt

Wie gelingt der Kuchen?

- Konsistenz: Gibt es ein Modell?

Kann man den Kuchen überhaupt backen?



James F. Allen:

Maintaining knowledge about temporal intervals

Commun. ACM, 1983

- Keine Darstellung von Zeitpunkten, sondern:
- Darstellung von **Zeitintervallen**
- **ohne** Absolutwerte (weder von wann bis wann noch wie lang)
- sondern: nur die **relative Lage** von Intervallen

Allensche Formeln:

$$F ::= (A \ r \ B) \mid \neg F \mid F_1 \vee F_2 \mid F_1 \wedge F_2$$

wobei

- A, B sind Intervallnamen
- r ist eine der Allenschen Basisrelationen

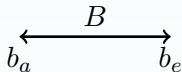
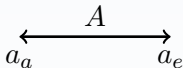
Allensche Formeln:

$$F ::= (A \ r \ B) \mid \neg F \mid F_1 \vee F_2 \mid F_1 \wedge F_2$$

wobei

- A, B sind Intervallnamen
- r ist eine der Allenschen Basisrelationen

Basisrelationen: Gegeben zwei nichtleere reellwertige Intervalle:

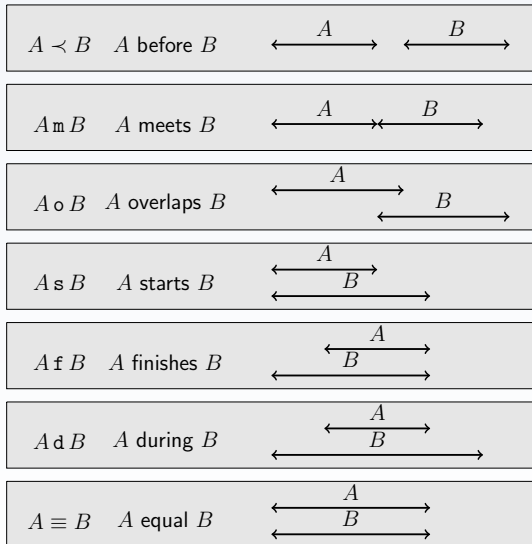


- Wie können A und B zueinander liegen?
- Wieviele Möglichkeiten gibt es?

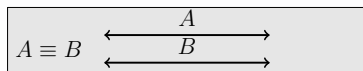
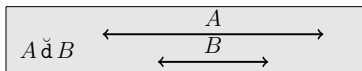
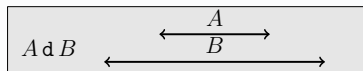
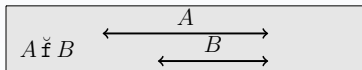
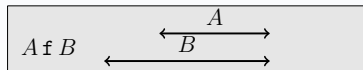
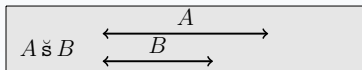
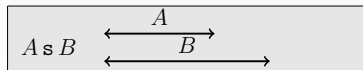
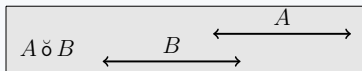
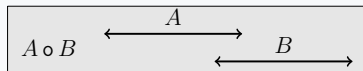
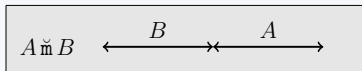
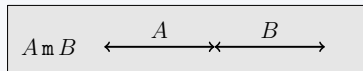
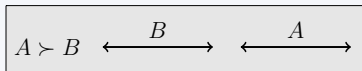
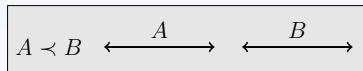
<i>Bedingung</i>	<i>Abkürzung</i>	<i>Bezeichnung</i>
$AZ < BA$	\prec	A before B
$AZ = BA$	m	A meets B
$AA < BA < AZ < BZ$	o	A overlaps B
$AA = BA < AZ < BZ$	s	A starts B
$BA < AA < AZ = BZ$	f	A finishes B
$BA < AA < AZ < BZ$	d	A during B
$BA = AA, AZ = BZ$	\equiv	A equal B

- und inverse Relationen (ohne \equiv)
- Inverse: \check{r} ist inverse Relation zu r
- Ausnahme (in der Schreibweise): \succ inverses zu \prec
- und $\equiv = \check{\equiv}$

Allensche Basisrelationen, Teil 1



Alle Allensche Basisrelationen



Allensche Basisrelationen

Die 13 Allenschen Basis-Relationen sind:

$$\mathcal{R} := \{\equiv, \prec, m, o, s, d, f, \succ, \check{m}, \check{o}, \check{s}, \check{d}, \check{f}\}.$$

Satz

Die Allenschen Basis-Relationen sind paarweise disjunkt, d.h.

$$A r_1 B \wedge A r_2 B \implies r_1 = r_2.$$

Schreibweise

$$A\{r_1, \dots, r_n\}B := (A r_1 B) \vee (A r_2 B) \dots \vee (A r_n B)$$

$A\{r_1, \dots, r_n\}B$ nennt man **atomares Allen-Constraint**

Beispiele



vor



Zutaten {<, m} TeigZub



direkt nach



TeigR {m̃} TeigZub



während,
aber vorher
endend



Sahne {s, d} BelagZub

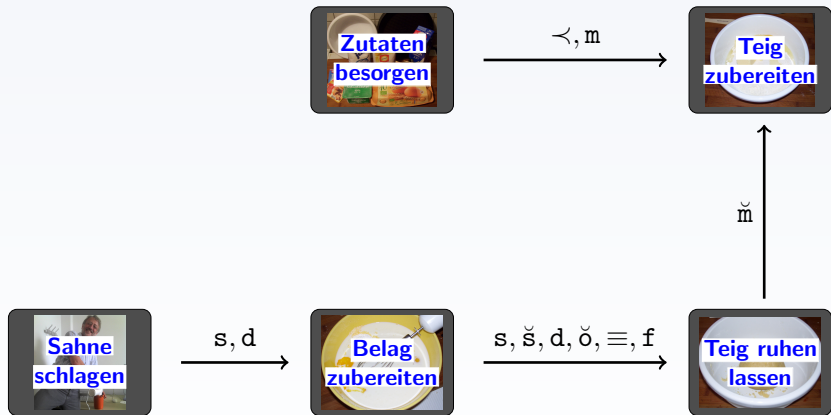


beginnt
während

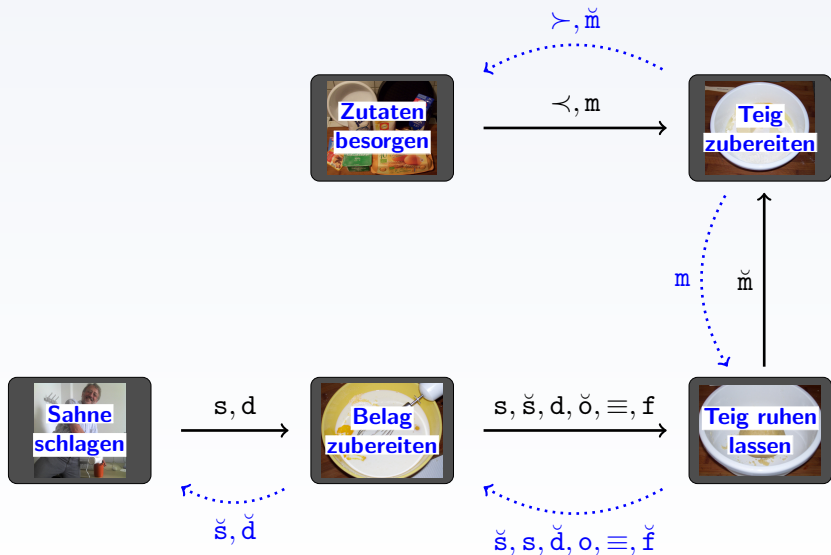


BelagZub {s, š, d, ö, ≡, f} TeigR

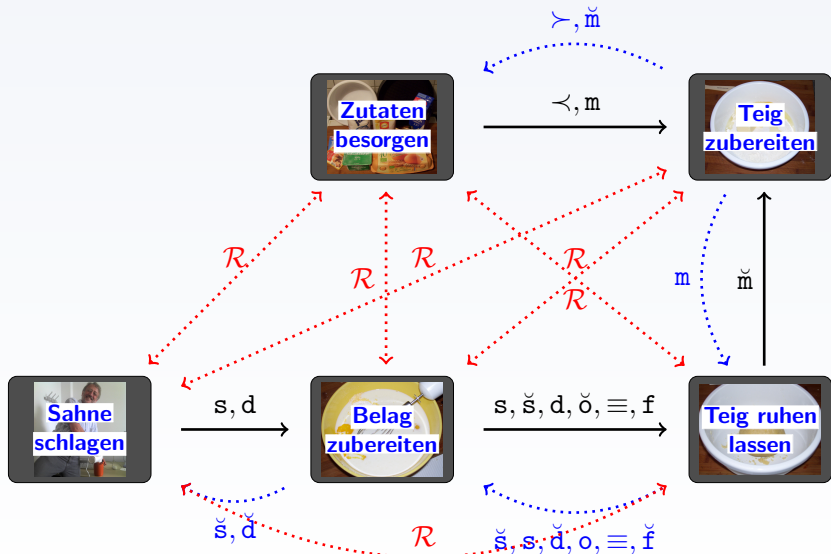
Beispiel als Constraintnetzwerk



Beispiel als Constraintnetzwerk



Beispiel als Constraintnetzwerk



Interpretation I :

bildet Intervallnamen auf Intervalle $[a, b]$ ab,
wobei $a, b \in \mathbb{R}$ und $a < b$.

Interpretation von atomaren Aussagen $A r B$:

Sei $I(A) = [AA, AZ]$ und $I(B) = [BA, BZ]$.

- $I(A < B) = 1$, gdw. $AZ < BA$
- $I(A = B) = 1$, gdw. $AZ = BA$
- $I(A \circ B) = 1$, gdw. $AA < BA$, $BA < AZ$ und $AZ < BZ$
- $I(A s B) = 1$, gdw. $AA = BA$ und $AZ < BZ$
- $I(A f B) = 1$, gdw. $AA > BA$ und $AZ = BZ$
- $I(A d B) = 1$, gdw. $AA > BA$ und $AZ < BZ$
- $I(A \equiv B) = 1$, gdw. $AA = BA$ und $AZ = BZ$
- $I(A \check{r}_0 B) = 1$, gdw. $I(B r_0 A) = 1$
- $I(A \succ B) = 1$, gdw. $I(B < A) = 1$

Interpretation von Allenschen Formeln:

- $I(F \wedge G) = 1$ gdw. $I(F) = 1$ und $I(G) = 1$
- $I(F \vee G) = 1$ gdw. $I(F) = 1$ oder $I(G) = 1$.
- $I(\neg F) = 1$ gdw. $I(F) = 0$
- $I(F \iff G) = 1$ gdw. $I(F) = I(G)$
- $I(F \Rightarrow G) = 1$ gdw. $I(F) = 0$ oder $I(G) = 1$

D.h.: wie üblich

Interpretation I ist ein **Modell** für F gdw. $I(F) = \text{True}$ gilt.

Eine Allensche Formel F ist:

- **widersprüchlich** (inkonsistent), wenn es **kein Modell** für F gibt.
- **allgemeingültig**, wenn jede Interpretation ein Modell für F ist.
- **erfüllbar**, wenn es **mindestens ein Modell** für F gibt.

Zwei Formeln F und G sind **äquivalent** gdw. $\forall I : I(F) = I(G)$

Semantische Folgerung: $G \models F$ gdw. $\forall I : I(G) = 1 \Rightarrow I(F) = 1$

- Zur Erinnerung: $A \ S \ B$ mit $S \subseteq \mathcal{R}$ nennen wir **atomares Allen-Constraint**
- Z.B.: Statt $A \prec B \vee A \ s \ B \vee A \ f \ B$ schreiben wir $A \ \{\prec, s, f\} \ B$
- Beachte: Es gibt 2^{13} solche Mengen S .
- Auch erlaubt: $A \ \emptyset \ B$, Semantik: $I(A \ \emptyset \ B) = 0$.
- $A \ \mathcal{R} \ B$ bedeutet: alles ist möglich, $I(A \ \mathcal{R} \ B) = 1$.

Vereinfachungsregeln für Allensche Formeln

- Ein atomare Aussage der Form $A \ r \ A$ kann man immer vereinfachen zu 0, 1:
 - $A \ r \ A \rightarrow 0$, wenn $r \neq \equiv$ und
 - $A \equiv A \rightarrow 1$.
- Negationszeichen kann man nach innen schieben.
- Eine Formel $\neg(A \ R \ B)$ kann man zu $A \ (\mathcal{R} \setminus R) \ B$ umformen.
- Unterformeln der Form $A \ R_1 \ B \wedge A \ R_2 \ B$ kann man durch $A \ (R_1 \cap R_2) \ B$ ersetzen.
- Unterformeln der Form $A \ R_1 \ B \vee A \ R_2 \ B$ kann man durch $A \ (R_1 \cup R_2) \ B$ ersetzen.
- atomare Formeln der Form $A \ \emptyset \ B$ kann man durch 0 ersetzen.
- atomare Formeln der Form $A \ \mathcal{R} \ B$ kann man durch 1 ersetzen.
- Alle aussagenlogischen Umformungen sind erlaubt.

Theorem

Jede Vereinfachungsregel für Allensche Formeln erhält die Äquivalenz, d.h. wenn $F \rightarrow F'$, dann sind F und F' äquivalente Formeln.

Beweis: Verwende die Semantik

Mit den Vereinfachungen kann jede Allensche Formel umgeformt werden in ein

Disjunktives Allen-Constraint

- **(konjunktives) Allen-Constraint:**

Eine Konjunktion von atomaren Allen-Constraints:

$$A_1 S_1 A_2 \wedge \dots \wedge A_n S_n A_n$$

- **Disjunktives Allen-Constraint:**

Disjunktion von (konjunktiven) Allen-Constraints

- Eingabe: Allen-Constraint
- Ausgabe: Weitere Beziehungen die daraus folgen, bzw. 0 (Widerspruch) oder 1 (Tautologie)
- Es reicht im Grunde: Konjunktive Allen-Constraints
- Bei disjunktiven Allen-Constraints: bearbeite die Allen-Constraints unabhängig und füge dann zusammen.

Wesentliche Regel: „Transitivitätsregel“

- Aus $A \prec B \wedge B \prec C$ kann man $A \prec C$ folgern.
- Aus $A \prec B \wedge C \prec B$ kann man nichts neues über die Beziehung zwischen A und C folgern (alles ist möglich)

Wie folgert man genau?

- Basisrelationen $r_1, r_2: A r_1 B \wedge B r_2 C$.

Man braucht die Komposition $(r_1 \circ r_2)$, als kleinste Menge mit: $A r_1 B \wedge B r_2 C \models A(r_1 \circ r_2)C$.

Beachte: $(r_1 \circ r_2)$ ist nicht unbedingt eine Basisrelation

- $R_1, R_2 \subseteq \mathcal{R}: A R_1 B \wedge B R_2 C$.

Komposition der Mengen: Sei $R_1 \circ R_2$ gerade die (kleinste) Menge mit: $A R_1 B \wedge B R_2 C \models A(R_1 \circ R_2)C$.

Kompositionsmatrix

	λ	γ	d	d̂	o	ô	≡	≡̂	s	ŝ	f	f̂
λ	λ	ℛ	λ d s	λ	λ	λ d s	λ	λ d s	λ	λ	λ d s	λ
γ	ℛ	γ	γ d f	γ	γ d f	γ	γ d f	γ	γ d f	γ	γ	γ
d	λ	γ	d	ℛ	λ d s	γ d f	λ	γ	d	γ d f	d	λ d s
d̂	λ d̂ ô Ĥ K̂ ≡̂	γ d̂ ô ŝ ≡̂	ℛ λ γ ≡̂	d̂	o d̂ Ĥ K̂	ô d̂ ŝ ≡̂	o d̂ Ĥ K̂	ô d̂ ŝ ≡̂	o d̂ Ĥ K̂	d̂	ô d̂ ŝ ≡̂	d̂
o	λ	γ d̂ ô ŝ ≡̂	o d s	λ d̂ ô f̂	λ o ≡̂	λ γ ≡̂	λ	ô d̂ ŝ ≡̂	o	d̂ Ĥ K̂	d s o	λ o ≡̂
ô	λ d̂ ô Ĥ K̂ ≡̂	γ	ô d̂ f̂	γ d̂ ô ŝ ≡̂	λ γ ≡̂	γ ô d̂ ŝ ≡̂	o d̂ Ĥ K̂	γ	ô d̂ f̂	γ ô d̂ ŝ ≡̂	ô	ô d̂ ŝ ≡̂
≡	λ	γ d̂ ô ŝ ≡̂	o d s	λ	λ	o d s	λ	≡̂ Ĥ K̂	≡̂	≡̂	d s o	λ
≡̂	λ d̂ ô Ĥ K̂ ≡̂	γ	ô d̂ f̂	γ	ô d̂ f̂	γ	≡̂ ŝ ≡̂	γ	d̂ Ĥ ô	γ	≡̂	≡̂
s	λ	γ	d	λ d̂ ô Ĥ K̂ ≡̂	λ o ≡̂	ô d̂ f̂	λ	≡̂	s	≡̂ ŝ ≡̂	d	λ o ≡̂
ŝ	λ d̂ ô Ĥ K̂ ≡̂	γ	ô d̂ f̂	d̂	o d̂ Ĥ K̂	ô	o d̂ Ĥ K̂	≡̂	≡̂ ŝ ≡̂	≡̂ ŝ ≡̂	ô	d̂
f	λ	γ	d	γ d̂ ô ŝ ≡̂	o d s	γ ô d̂ ŝ ≡̂	≡̂	γ	d	γ ô d̂ ŝ ≡̂	f	≡̂ Ĥ K̂
f̂	λ	γ d̂ ô ŝ ≡̂	o d s	d̂	o	ô d̂ ŝ ≡̂	≡̂	ô d̂ ŝ ≡̂	o	d̂	≡̂ Ĥ K̂	f̂

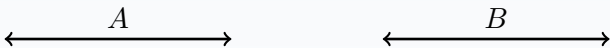
Nur 12 × 12-Matrix, da:

$$ro \equiv = r = \equiv or$$

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$

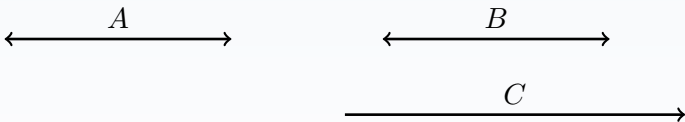


Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$

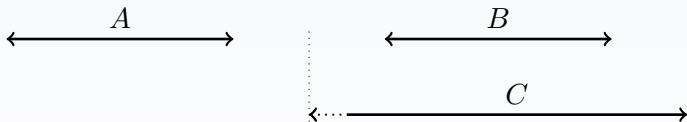


Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$

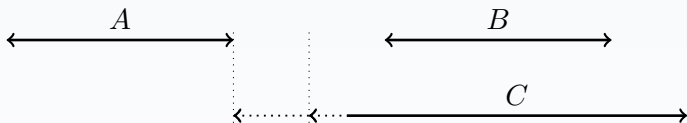


Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$



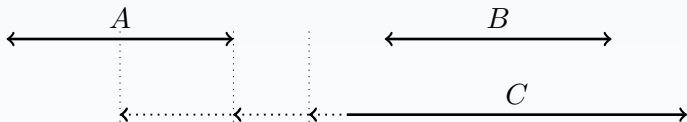
Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$



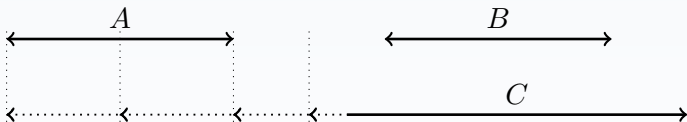
Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$



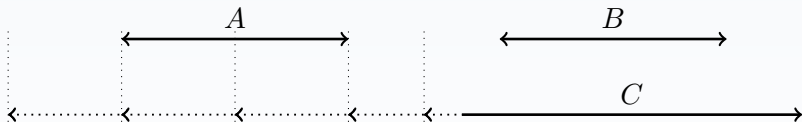
Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Kompositionsmatrix

Die Einträge muss/kann man per Hand ausrechnen.

Beispiel: $\prec \circ d$

Betrachte alle Möglichen Lagen für $A \prec B \wedge B d C$



Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Beispiel: Aus $A \{m, d\} B \wedge B \{f, d\} C$ kann man schließen

$$\begin{aligned} & A (m \circ f \cup m \circ d \cup d \circ f \cup d \circ d) C \\ = & A \{d, s, o\} \cup \{d, s, o\} \cup \{d\} \cup \{d\} C \\ = & A \{d, s, o\} C \end{aligned}$$

Allgemein gilt:

Satz

Seien $r_1, \dots, r_k, r'_1, \dots, r'_k$ Allensche Basisrelationen. Dann gilt

$$\{r_1, \dots, r_k\} \circ \{r'_1, \dots, r'_k\} = \bigcup \{r_i \circ r'_j \mid i = 1, \dots, k, j = 1, \dots, k'\}$$

Inversion für Mengen von Basisrelationen

Sei $S = \{r_1, \dots, r_k\} \subseteq \mathcal{R}$. Dann sei

$$\check{S} = \{\check{r}_1, \dots, \check{r}_k\}.$$

Desweiteren sei $\check{\check{r}} = r$

Damit gilt:

Satz

Für $S \subseteq \mathcal{R}$ gilt: $A S B$ und $B \check{S} A$ sind äquivalente Allensche Formeln.

Satz

$$\check{.(r_1 \circ r_2)} = \check{r}_2 \circ \check{r}_1.$$

Eingabe: Konjunktives Allen-Constraint

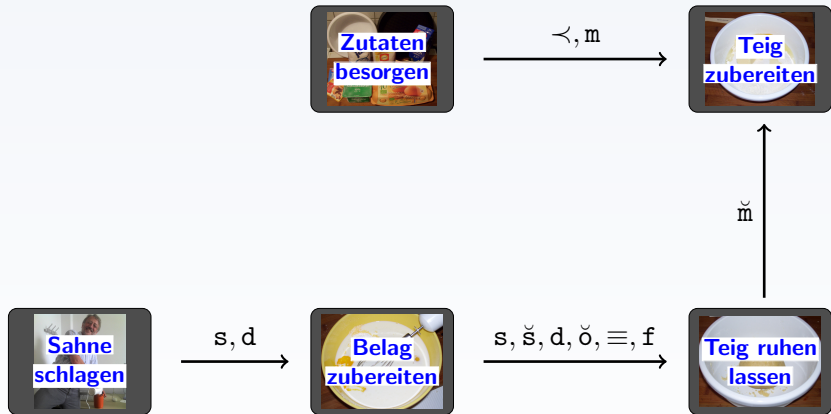
Ausgabe: Allenscher Abschluss

Verfahren: Berechne Fixpunkt bezüglich der Regeln (auf Subformeln):

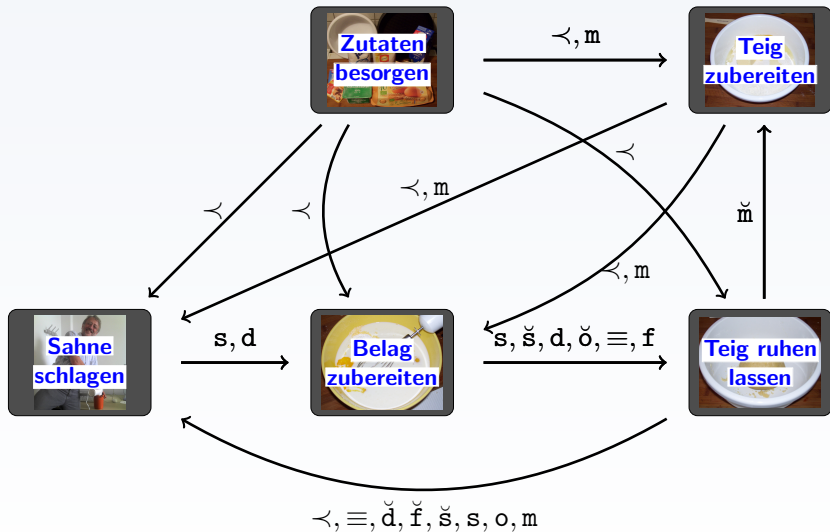
- Vereinfachungen: (\rightarrow bedeutet „ersetze“)
 - $A R_1 B \wedge A R_2 B \rightarrow A (R_1 \cap R_2) B$
 - $A \emptyset B \rightarrow \text{False}$
 - $A \mathcal{R} B \rightarrow \text{True}$
 - $A R A \rightarrow \text{False}$, wenn $\equiv \notin R$.
 - $A R A \rightarrow \text{True}$, wenn $\equiv \in R$.
- Folgerungen: (\rightsquigarrow bedeutet „füge hinzu“)
 - $A R B \rightsquigarrow B \check{R} A$, wobei $\check{R} := \{\check{r}_1, \dots, \check{r}_n\}$ für $R = \{r_1, \dots, r_n\}$
 - $A R_1 B \wedge B R_2 C \rightsquigarrow A (R_1 \circ R_2) C$.
- und übliche aussagenlogische Umformungen

- Für konjunktive Allensche Constraints: Wende die Regeln des Allenschen Kalküls solange an, bis sich keine neuen Beziehungen mehr herleiten lassen (Fixpunkt)
- Disjunktive Constraints: Wende Fixpunktiteration auf jede Komponente an, und vereinfache anschließend
 - Komponente = 1: Disjunktiver Constraints ist äquivalent zu 1
 - Komponente = 0: Kann gestrichen werden
 - Alle Komponenten = 0: Disjunktiver Constraint widersprüchlich (Inkonsistenz)

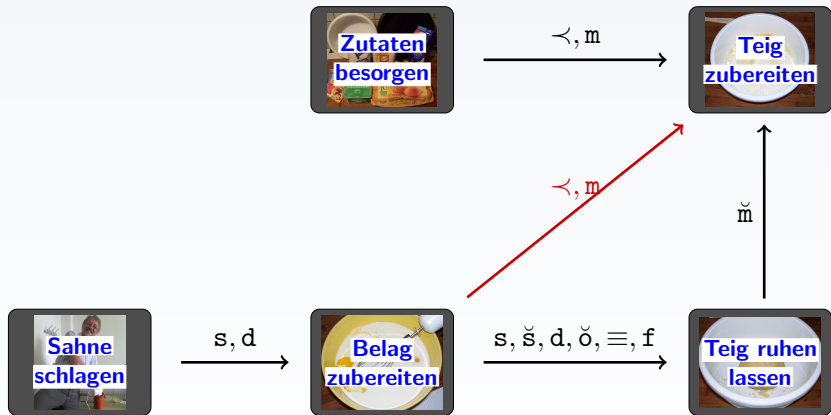
Beispiel



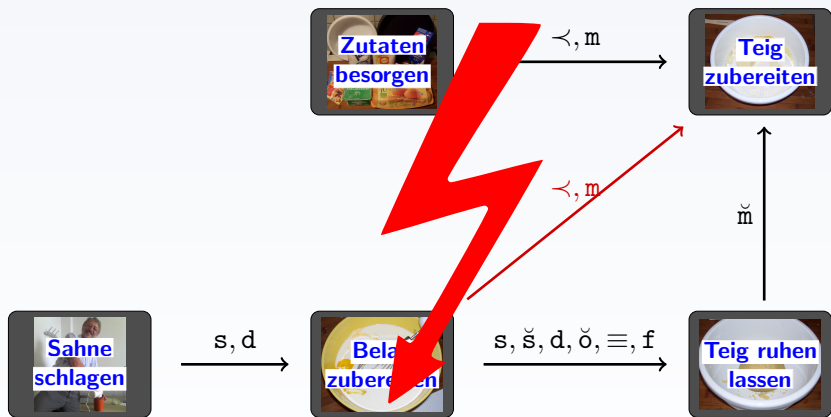
Beispiel



Beispiel (2)



Beispiel (2)



Wir sagen, der Allen-Kalkül ist

- **korrekt**, wenn bei $F \rightarrow F'$ stets gilt: F und F' sind äquivalente Formeln
- **herleitungs-vollständig**, wenn er für jedes konjunktive Constraint alle semantisch folgerbaren Einzel-Relationen herleiten kann.
- **widerspruchs-vollständig**, wenn er für jedes konjunktive Constraint herausfinden kann, ob es widersprüchlich ist (Herleitung der 0)

- Wie aufwändig ist die Berechnung des Abschlusses der Allenschen Relationen?
- Ist der Allen-Kalkül korrekt?
- Ist die Berechnung herleitungs- bzw- widerspruchs-vollständig?
- Was ist die Komplexität der Logik und der Herleitungsbeziehung, evtl. für eingeschränkte Eingabeformeln?
- Wie kann man den Allenschen Kalkül für aussagenlogische Kombinationen von Intervallformeln verwenden?

Implementierung der Allen-Vervollständigung

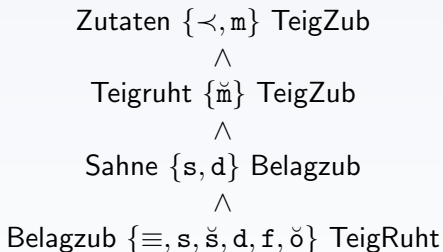
- Wesentliche Regel: Transitivitätsregel
 $A R_1 B \wedge B R_2 C \rightarrow A R_1 \circ R_2 C.$
- Konjunktive Allen-Constraints (schon zusammengefasst, Intervalle A_1, \dots, A_n):

$$\bigwedge_{i,j \in \{1, \dots, n\}} A_i R_{i,j} A_j$$

Nicht vorhandene Relation werden auf \mathcal{R} gesetzt.

- Abschluss kann mit einer $n \times n$ -Tabelle gemacht werden
- Sobald \emptyset irgendwo auftaucht, kann man abbrechen
- Ähnlich zum Warshall-Algorithmus
- Bei disjunktiven Allen-Constraints: bearbeite die Allen-Constraints separat und fasse dann zusammen.

Beispiel für das Eingabearray



$R_{i,j}$	(1) Zutaten	(2) Teigzub	(3) Teigruht	(4) Sahne	(5) Belagzub
(1) Zutaten	$\{\equiv\}$	$\{\prec, m\}$	\mathcal{R}	\mathcal{R}	\mathcal{R}
(2) Teigzub	$\{\succ, \check{m}\}$	$\{\equiv\}$	$\{m\}$	\mathcal{R}	\mathcal{R}
(3) Teigruht	\mathcal{R}	$\{\check{m}\}$	$\{\equiv\}$	\mathcal{R}	$\{\equiv, \check{d}, \check{f}, s, \check{s}, o\}$
(4) Sahne	\mathcal{R}	\mathcal{R}	\mathcal{R}	$\{\equiv\}$	$\{d, s\}$
(5) Belagzub	\mathcal{R}	\mathcal{R}	$\{\equiv, s, \check{s}, d, f, \check{o}\}$	$\{\check{d}, \check{s}\}$	$\{\equiv\}$

Algorithmus Allenscher Abschluss, Variante 1

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Algorithmus:

repeat

 change := False;

for $i := 1$ **to** n **do**

for $j := 1$ **to** n **do**

for $k := 1$ **to** n **do**

$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j});$

if $R_{i,j} \neq R'$ **then**

$R_{i,j} := R';$

 change := True;

endif

endfor

endfor

endfor

until change=False

Algorithmus 1

Algorithmus Allenscher Abschluss, Variante 1

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Algorithmus:

repeat

 change := False;

for $i := 1$ **to** n **do**

for $j := 1$ **to** n **do**

for $k := 1$ **to** n **do**

$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j});$

if $R_{i,j} \neq R'$ **then**

$R_{i,j} := R';$

 change := True;

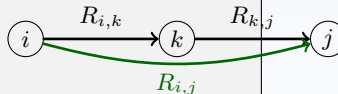
endif

endfor

endfor

endfor

until change=False



$$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$$

entspricht gerade

$$A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,j} A_j$$

$$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$$

entspricht gerade

$$\rightarrow A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,j} A_j \\ \rightarrow A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,k} \circ R_{k,j} A_j \wedge A_i R_{i,j} A_j$$

$$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$$

entspricht gerade

$$\begin{aligned} & A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,j} A_j \\ \rightarrow & A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,k} \circ R_{k,j} A_j \wedge A_i R_{i,j} A_j \end{aligned}$$

$$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$$

entspricht gerade

$$\begin{aligned} & A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,j} A_j \\ \rightarrow & A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i R_{i,k} \circ R_{k,j} A_j \wedge A_i R_{i,j} A_j \\ \rightarrow & A_i R_{i,k} A_k \wedge A_k R_{k,j} A_j \wedge A_i (R_{i,k} \circ R_{k,j}) \cap R_{i,j} A_j \end{aligned}$$

Eigenschaften Algorithmus 1

- Ähnlich zu Warshall-Algorithmus, aber iteriert (notwendig!)
- solange bis Fixpunkt erreicht ist
- Korrekt: Offensichtlich

Laufzeit: Im worst-case $O(n^5)$

Begründung:

- 3 for-Schleifen: $O(n^3)$
- repeat-Schleife: Im schlechtesten Fall wird ein $R_{i,j}$ um eins verkleinert
- pro $R_{i,j}$ maximal 13 Verkleinerungen
- Es gibt n^2 Mengen $R_{i,j}$
- Daher: repeat-Schleife wird maximal $O(n^2)$ mal durchlaufen
- ergibt: $O(n^5)$

Algorithmus Allenscher Abschluss, Variante 2

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Algorithmus:

queue := $\{(i, k, j) \mid 1 \leq i \leq n, 1 \leq k \leq n, 1 \leq j \leq n\}$;

while queue $\neq \emptyset$ **do**

 Wähle und entferne Tripel (i, k, j) aus queue;

$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$;

if $R_{i,j} \neq R'$ **then**

$R_{i,j} := R$;

 queue := queue ++ $\{(i, j, m) \mid 1 \leq m \leq n\}$ ++ $\{(m, i, j) \mid 1 \leq m \leq n\}$

endif

endwhile

Algorithmus Allenscher Abschluss, Variante 2

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Algorithmus:

queue := $\{(i, k, j) \mid 1 \leq i \leq n, 1 \leq k \leq n, 1 \leq j \leq n\}$;

while queue $\neq \emptyset$ **do**

 Wähle und entferne Tripel (i, k, j) aus queue;

$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$;

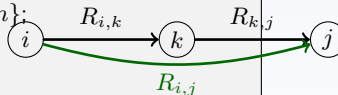
if $R_{i,j} \neq R'$ **then**

$R_{i,j} := R$;

 queue := queue ++ $\{(i, j, m) \mid 1 \leq m \leq n\}$ ++ $\{(m, i, j) \mid 1 \leq m \leq n\}$

endif

endwhile



Algorithmus 2

Algorithmus Allenscher Abschluss, Variante 2

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Algorithmus:

queue := $\{(i, k, j) \mid 1 \leq i \leq n, 1 \leq k \leq n, 1 \leq j \leq n\}$;

while queue $\neq \emptyset$ **do**

Wähle und entferne Tripel (i, k, j) aus queue;

$R' := R_{i,j} \cap (R_{i,k} \circ R_{k,j})$;

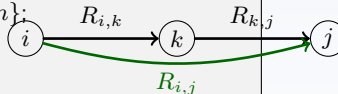
if $R_{i,j} \neq R'$ **then**

$R_{i,j} := R$;

queue := queue ++ $\{(i, j, m) \mid 1 \leq m \leq R_{j,m}\}$ ++ $\{(m, i, j) \mid 1 \leq m \leq R_{m,i}\}$

endif

endwhile



Eigenschaften Algorithmus 2

Korrektheit: Bei Änderung von $R_{i,j}$ werden alle Nachbarn, die evtl. neu berechnet werden müssen, in queue eingefügt

Eigenschaften Algorithmus 2

Korrektheit: Bei Änderung von $R_{i,j}$ werden alle Nachbarn, die evtl. neu berechnet werden müssen, in queue eingefügt

Laufzeit:

- Am Anfang: queue enthält n^3 Tripel
- while-Schleife entfernt pro Durchlauf ein Element aus queue
- Einfügen in queue in der Summe:
 - $R_{i,j}$ kann höchstens 13 mal verändert werden.
 - D.h. höchstens $n^2 * 13$ mal wird eingefügt
 - Einmal einfügen: $2 * n$ Tripel werden hinzugefügt

Insgesamt: Es werden höchstens $13 * 2 * n * n^2$ Tripel zu queue hinzugefügt

- Ergibt $O(n^3)$ Durchläufe der while-Schleife
(von denen maximal $O(n^2)$ Durchläufe $O(n)$ Laufzeit verbrauchen und die restlichen $O(n)$ in konstanter Laufzeit laufen)

Eigenschaften Algorithmus 2

Korrektheit: Bei Änderung von $R_{i,j}$ werden alle Nachbarn, die evtl. neu berechnet werden müssen, in queue eingefügt

Laufzeit:

- Am Anfang: queue enthält n^3 Tripel
- while-Schleife entfernt pro Durchlauf ein Element aus queue
- Einfügen in queue in der Summe:
 - $R_{i,j}$ kann höchstens 13 mal verändert werden.
 - D.h. höchstens $n^2 * 13$ mal wird eingefügt
 - Einmal einfügen: $2 * n$ Tripel werden hinzugefügt

Insgesamt: Es werden höchstens $13 * 2 * n * n^2$ Tripel zu queue hinzugefügt

- Ergibt $O(n^3)$ Durchläufe der while-Schleife
(von denen maximal $O(n^2)$ Durchläufe $O(n)$ Laufzeit verbrauchen und die restlichen $O(n)$ in konstanter Laufzeit laufen)

Algorithmus 2 hat worst-case-Laufzeit $O(n^3)$

Korrektheit

Der Allensche Kalkül ist **korrekt**, d.h. wenn $F \rightarrow F'$, dann sind F und F' äquivalente Formeln

Beweis (Skizze): Verwende die Semantik

- Aussagenlogische Umformungen: klar
- $A R_1 B \wedge A R_2 B$ ist äquivalent zu $A (R_1 \cap R_2) B$:

Sei $R_1 = \{r_1, \dots, r_k\}$, $R_2 = \{r'_1, \dots, r'_{k'}\}$.

$$\begin{aligned} & A R_1 B \wedge A R_2 B \\ = & (\bigvee A r_i B) \wedge (\bigvee A r'_{i'} B) \\ \sim & \bigvee \{(A r_i B) \wedge (A r'_{i'} B) \mid 1 \leq i \leq k, 1 \leq i' \leq k'\} \text{ (ausmultiplizieren)} \\ \sim & \bigvee \{(A r_i B) \wedge (A r'_{i'} B) \mid 1 \leq i \leq k, 1 \leq i' \leq k', r_i = r'_{i'}\} \text{ (Basisrelationen disjunkt)} \\ = & A (R_1 \cap R_2) B \end{aligned}$$

- $A \emptyset B \sim 0$ und $A \mathcal{R} B \sim 1$ (klar)

Beweis (Fortsetzung)

- $A R A$ ist äquivalent zu 0, wenn $\equiv \notin R$ und

$A R A$ ist äquivalent zu 1, wenn $\equiv \in R$:

Jede Interpretation bildet $I(A)$ eindeutig auf ein Intervall ab.

- Transitivitätsregel:

Basisrelationen: Man muss die Korrektheit der Matrix prüfen

Für mehrelementige Mengen:

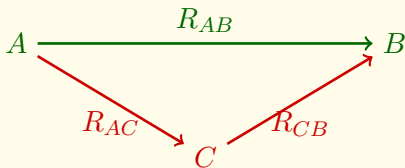
$$\begin{aligned} & A \{r_1, \dots, r_k\} B \wedge B \{r'_1, \dots, r'_{k'}\} C \\ = & (A r_1 B \vee \dots \vee A r_k B) \wedge (B r'_1 C \vee \dots \vee B r'_{k'} C) \text{ (ausmultiplizieren)} \\ \sim & \bigvee \{(A r_i B \wedge B r'_{i'} C) \mid 1 \leq i \leq k, 1 \leq i' \leq k'\} \text{ (Basis)} \\ \sim & \bigvee \{(A r_i B \wedge B r'_{i'} C \wedge A r_i \circ r'_{i'} C) \mid 1 \leq i \leq k, 1 \leq i' \leq k'\} \\ \sim & A \{r_1, \dots, r_k\} B \wedge B \{r'_1, \dots, r'_{k'}\} C \wedge \bigvee \{(A r_i \circ r'_{i'} C) \mid 1 \leq i \leq k, 1 \leq i' \leq k'\} \\ = & A \{r_1, \dots, r_k\} B \wedge B \{r'_1, \dots, r'_{k'}\} C \wedge A \{r_1, \dots, r_k\} \circ \{r'_1, \dots, r'_{k'}\} C \end{aligned}$$

Partielle Vollständigkeit

Der Allensche Kalkül ist vollständig in eingeschränktem Sinn:

Satz (Pfadkonsistenz)

Der Allensche Abschluss ist 3-konsistent:



Jede Belegung I der Intervalle A und B mit $I(A \ R_{AB} \ B) = \text{True}$ kann auf das Intervall C erweitert werden, so dass $I(A \ R_{BC} \ C) = \text{True} = I(C \ R_{AC} \ B)$.

Es gilt **nicht** (globale Konsistenz):

Jede Belegung von k Knoten kann auf $k + 1$ Knoten unter Erhaltung der Erfüllbarkeit erweitert werden

Leider gilt:

Theorem

Der Allensche Kalkül ist nicht herleitungs-vollständig.

Beweis: Gegenbeispiel: Für den Allenschen Constraint:

$$D \{o\} B \wedge D \{s, m\} C \wedge D \{s, m\} A \wedge A \{d, \check{d}\} B \wedge C \{d, \check{d}\} B$$

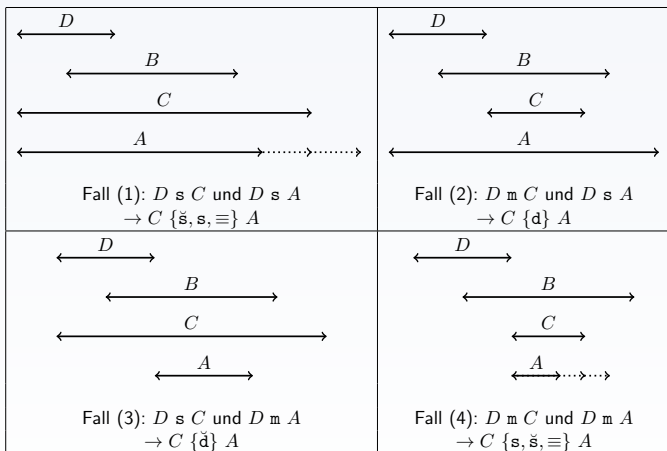
ist der Allensche Abschluss:

$$D \{o\} B \wedge D \{s, m\} C \wedge D \{s, m\} A \wedge A \{d, \check{d}\} B \wedge C \{d, \check{d}\} B \\ \wedge C \{s, \check{s}, \equiv, o, \check{o}, d, \check{d}, f, \check{f}\} A$$

Aber in diesem Fall ist $C \{f, \check{f}, o, \check{o}\} A$ nicht möglich (nächste Folie)

Beweis (Fortsetzung)

- Die Lage von B zu D ist eindeutig.
- Möglichkeiten wie A zu D und C zu D : 4 Fälle



$C \{\check{f}, \check{f}, o, \check{o}\} A$ nicht möglich!

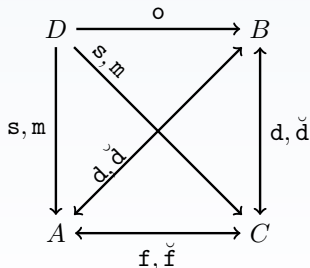
Unvollständigkeit des Allen-Kalküls (2)

Ebenso gilt:

Theorem

Der Allensche Kalkül ist nicht widerlegungsvollständig.

Beweis: Gegenbeispiel: Leichte Abwandlung des Beispiels davor
Füge $A \{f, \check{f}\} C$ hinzu, d.h. man erhält das Constraintnetzwerk:



Allenscher Abschluss: Verändert das Netzwerk nicht, aber es ist widersprüchlich!

Frage: Ist Allen-Constraint F widersprüchlich?

- Abschluss = 0, dann JA
- Abschluss = 1, dann NEIN
- Abschluss weder 0 noch 1: man weiß nichts

Frage: Ist Allen-Constraint F erfüllbar?

- Abschluss = 0, dann NEIN
- Abschluss = 1, dann JA (Tautologie)
- Abschluss weder 0 noch 1: man weiß nichts

Definition

Ein Allensches Constraint nennt man **eindeutig**, wenn für alle Paare A, B von Intervallkonstanten gilt: Das Constraint enthält genau eine Beziehung $A r B$, wobei r eine der dreizehn Basisrelationen ist.

Es gilt:

Satz

Der Allensche Abschluss eines eindeutigen Allenschen Constraints F ist entweder \emptyset , oder wiederum F .

Beweis: Jede Transitivitätsregelanwendung leitet \emptyset her, oder lässt Eintrag unverändert.

Satz (Valdés-Pérez, 1987)

Ein eindeutiges Allensches Constraint ist erfüllbar, gdw. der Allensche Kalkül bei Vervollständigung das Constraint nicht verändert, d.h. wenn es ein Fixpunkt ist.

Beweisidee: Zeige, wenn Allen-Kalkül kein Widerspruch entdeckt, dann ist Constraint erfüllbar.

Es gibt dann eine totale Ordnung der Intervallenden

Korollar

Auf eindeutigen Allen-Constraints ist der Allen-Kalkül korrekt und vollständig

Zu jedem Allenschen Constraint C kann man die **Menge aller zugehörigen eindeutigen Allenschen Constraints** D definieren, wobei gelten muss:

Wenn $A r B$ in D vorkommt und $A R B$ in C , dann gilt $r \in R$.

Lemma

Ein Allen-Constraint ist erfüllbar, gdw. es ein zugehöriges eindeutiges Constraint gibt, das erfüllbar ist.

Beweis: Klar

Vollständiges Verfahren

Algorithmus Erfüllbarkeitstest für konjunktive Allensche Constraints

Eingabe: $(n \times n)$ -Array R , mit Einträgen $R_{i,j} \subseteq \mathcal{R}$

Ausgabe: True (Widerspruch) oder False (erfüllbar)

function AllenSAT(R):

$R' :=$ AllenAbschluss(R);

if $\exists R'_{i,j}$ mit $R'_{i,j} = \emptyset$ **then return** True **endif**; // Widerspruch

if $\forall R'_{i,j}$ gilt: $|R'_{i,j}| = 1$ **then return** False // eindeutig und erfüllbar
else

wähle $R'_{i,j}$ mit $R'_{i,j} = \{r_1, r_2, \dots\}$;

$R^l := R'$; $R^l_{i,j} := \{r_1\}$; // kopiere R' und setze (i, j) auf r_1

$R^r := R'$; $R^r_{i,j} := R'_{i,j} \setminus \{r_1\}$; // kopiere R' und setze (i, j) auf r_2, \dots

return (ASAT(R^l) \wedge ASAT(R^r));

endif

Der Algorithmus ist korrekt und vollständig. Die Laufzeit ist im worst-case **exponentiell**. Mittlere Verzweigungsrate: 6,5

Satz

Das Erfüllbarkeitsproblem für konjunktive Allenschen Constraints ist **\mathcal{NP} -vollständig**.

Beweis:

Problem ist in \mathcal{NP} :

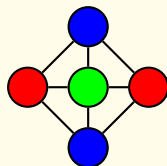
- Rate lineare Reihenfolge der Intervallanfänge und -enden
- D.h. Ordnung auf allen X_a, X_e für alle Intervalle X
- Verifiziere ob Reihenfolge das Constraint erfüllt
- Verifikation geht in Polynomialzeit

\mathcal{NP} -Härte:

Reduktion von 3-Färbbarkeit auf
Erfüllbarkeit von Allen-Constraints

3-Färbbarkeit:

Kann man die Knoten eines ungerichteten Graphen mit drei Farben färben, so dass benachbarte Knoten stets verschiedene Farben haben?



Beweis (Fortsetzung)

Für $G = (V, E)$ erzeuge:

Beweis (Fortsetzung)

Für $G = (V, E)$ erzeuge:

- (Rot m Gruen) \wedge (Gruen m Blau)



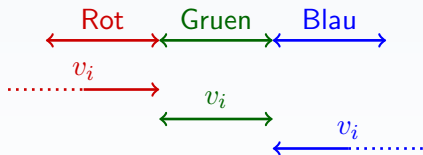
Beweis (Fortsetzung)

Für $G = (V, E)$ erzeuge:

- $(\text{Rot} \text{ m Gruen}) \wedge (\text{Gruen} \text{ m Blau})$



- Für die Knoten: $\forall v_i \in V : v_i \{m, \equiv, \checkmark\} \text{ Gruen}$



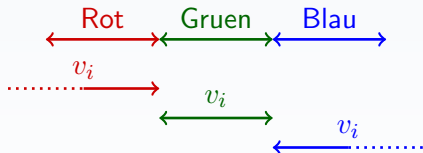
Beweis (Fortsetzung)

Für $G = (V, E)$ erzeuge:

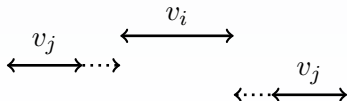
- (Rot m Gruen) \wedge (Gruen m Blau)



- Für die Knoten: $\forall v_i \in V : v_i \{m, \equiv, \check{m}\}$ Gruen



- Für die Kanten: $\forall (v_i, v_j) \in E : v_i \{m, \check{m}, \prec, \succ\} v_j$



Daher gilt: Der Graph ist dreifärbbar, gdw. die Allenschen Relationen erfüllbar sind. Die Zuordnung ist:

- v_i hat Farbe grün gdw. $v_i \equiv \text{Gruen}$
- v_i hat Farbe rot gdw. $v_i \vDash \text{Gruen}$
- v_i hat Farbe blau gdw. $v_i \not\equiv \text{Gruen}$

Übersetzung ist in Polynomialzeit durchführbar, daher Erfüllbarkeit \mathcal{NP} -hart

- Jeder vollständige Algorithmus braucht Exponentialzeit.
(solange $\mathcal{P} \neq \mathcal{NP}$)
- Die polynomielle Allen-Vervollständigung **muss** unvollständig sein

Es gibt polynomielle, vollständige Verfahren für

Allensche Constraints mit **eingeschränkter Syntax**

Eine haben wir bereits gesehen:

- Eindeutige Allen-Constraints

Neue Variante:

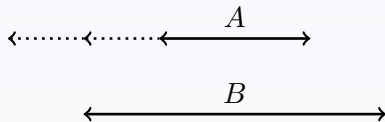
- Erlaube nur Allensche Relationen, so dass:
- Übersetzung in Bedingungen über die Endpunkte nur Konjunktionen von der Form $x < y$ oder $x = y$
- Dann gilt: Man braucht keine Fallunterscheidung

Passender Satz von Relationen:

- Alle Basisrelationen,
- $\{\underline{d}, \underline{o}, \underline{s}\}$, und $\{\underline{\ddot{o}}, \underline{f}, \underline{d}\}$ und deren Konverse. d.h. $\{\underline{\check{d}}, \underline{\check{o}}, \underline{\check{s}}\}$, und $\{\underline{o}, \underline{\check{f}}, \underline{\check{d}}\}$.

Varianten (3)

Z.B. $A \{d, o, s\} B$ als Ungleichung über den Endpunkten:



Wenn $A = [AA, AZ]$, $B = [BA, BZ]$, dann entspricht obige Relation gerade

$$AA < AZ, BA < BZ, AZ < BZ, BA < AZ$$

Varianten (4)

Auf solchen Constraints kann man Erfüllbarkeit in Polynomialzeit testen

- Transitiver Abschluss der Endpunktbeziehungen
- anschließend lineare Reihenfolge mit topologischem Sortieren

Es gilt aber sogar

Satz (Nebel, Bürckert, 1995)

Auf den so eingeschränkten Allen-Constraints ist der Allensche Kalkül korrekt und vollständig.

Diese spezielle Klasse lässt sich als **Grund-Hornklauseln** darstellen, d.h. Klauseln mit maximal einem positiven Literal.

Für Grund-Hornklauselmengen ist Erfüllbarkeit in polynomieller Zeit testbar.

Man hat Fakten in der Form $a < b$ und $c = d$, wobei a, b, c, d unbekannte Konstanten sind. Es gibt auch Hornklauseln, die von der Symmetrie und Transitivität stammen:

$$x < y \wedge y < z \Rightarrow x < z$$

$$x = y \wedge y = z \Rightarrow x = z$$

$$x = y \Rightarrow y = x$$

$$x < y \wedge y = z \Rightarrow x < z$$

$$x = y \wedge y < z \Rightarrow x < z$$

Hintergrund (2)

Man kann weitere Allensche Constraints zulassen, und behält die Vollständigkeit des Allen-Kalküls:

- Alle Constraints deren Übersetzung in Constraints über Endpunkten Hornklauseln ausschließlich mit Literalen $a \leq b$, $a = b$ und $\neg(a = b)$ erzeugt.
- Von den $2^{13} = 8192$ möglichen Beziehungen erfüllen 868 diese Eigenschaft

Man kann diese auch für die Fallunterscheidung des exponentiellen Verfahrens verwenden.

Vorteil: Kleinere mittlere Verzweigungsrate
(Statt 6,5 nur 2,533 (Nebel 1997))

Kuchen backen



Zutaten {<,m}	TeigZub,	Zutaten {<,m}	TeigRuht,	Zutaten {<,m}	BelZub,
Zutaten {<,m}	Sahne,	TeigRuht {m}	TeigForm,	TeigForm {<,m}	BelForm,
BelForm {<,m}	Backen,	Backen {f}	Heizen,	Sahne {s,d}	BelZub,
Heizen {S,o,>}	TeigRuht,	Heizen {<,m}	Kuehlen,	Kuehlen {<,m}	Entnehmen,
Zutaten {<,m}	Einfett,	Zutaten {<,m}	Heizen,	TeigZub {<,m}	TeigForm,
BelZub {<,m}	BelForm,	TeigZub {<,>,m,M}	BelZub,	Einfett {<,>,m,M}	BelZub,
Einfett {>,M}	TeigZub,	Einfett {<,m,M,>}	TeigForm,	Einfett {<,>,m,M}	BelForm,
Einfett {<,m}	TeigForm,	TeigZub {m}	TeigRuht,	Einfett {>,M}	Zutaten,
Backen {m}	Kuehlen,	BelZub {s,S,d,O,=,f}	TeigRuht,	Einfett {>}	Sahne

Kuchen backen



Zutaten {<,m}	TeigZub,	Zutaten {<,m}	TeigRuht,	Zutaten {<,m}	BelZub,
Zutaten {<,m}	Sahne,	TeigRuht {m}	TeigForm,	TeigForm {<,m}	BelForm,
BelForm {<,m}	Backen, Backen {f}	Heizen,	Heizen,	Sahne {s,d}	BelZub,
Heizen {S,o,>}	TeigRuht,	Heizen {<,m}	Kuehlen,	Kuehlen {<,m}	Entnehmen,
Zutaten {<,m}	Einfett,	Zutaten {<,m}	Heizen,	TeigZub {<,m}	TeigForm,
BelZub {<,m}	BelForm,	TeigZub {<,>,m,M}	BelZub,	Einfett {<,>,m,M}	BelZub,
Einfett {>,M}	TeigZub,	Einfett {<,m,M,>}	TeigForm,	Einfett {<,>,m,M}	BelForm,
Einfett {<,m}	TeigForm,	TeigZub {m}	TeigRuht,	Einfett {>,M}	Zutaten,
Backen {m}	Kuehlen,	BelZub {s,S,d,0,=,f}	TeigRuht,	Einfett {>}	Sahne

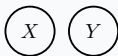
Allen-Programm:

Max. #Modelle in der Eingabe : 177.247.393.995.618.482.069.389.150.242.626.279.322.671.526.463.930.368
Max. #Modelle nach Allen-Abschluss: 5.898.240

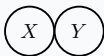
Anzahl Modelle : 1.536
Allenscher Abschluss genau? : True

Qualitatives räumliches Schließen

- Eindimensional: Genau die Allensche Intervalllogik
- Zweidimensional: Region-Connection-Calculus (RCC8),
(Randell, Cui & Cohn, 1992)



$X \text{ DC } Y$
„disconnected“



$X \text{ EC } Y$
„externally
connected“



$X \text{ TPP } Y$
„tangential
proper part“



$X \text{ NTPP } Y$
„non-tangential
proper part“



$X \text{ PO } Y$
„partially
overlapping“



$X \text{ EC } Y$
„equal“



$X \text{ TPPi } Y$
„tangential
proper part inverse“



$X \text{ NTPPi } Y$
„non-tangential
proper part inverse“