

Logikbasierte Systeme der Wissensverarbeitung

Allens Zeitintervallogik

Prof. Dr. M. Schmidt-Schauß

SoSe 2023

- Darstellung und Inferenzen für zeitliche Zusammenhänge
- Viele verschiedene Logiken
- z.B. Modallogiken und Temporallogiken.
Diese sprechen über Ereignisse in der Zukunft /
Vergangenheit und haben Existenzquantoren
und haben oft exakte Zeitdauern.

Wir betrachten die

Allensche Intervall-Logik

Beispiel: Käse-Sahne-Kuchen backen



Zutaten

besorgen

Beispiel: Käse-Sahne-Kuchen backen

Zutaten

besorgen



Zutaten

besorgen

Teig

zubereiten

Zutaten

besorgen

Teig

zubereiten

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

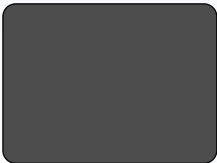
Belag
zubereiten

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten



Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen



Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Beispiel: Käse-Sahne-Kuchen backen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

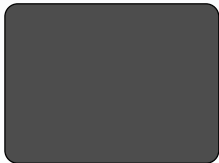
Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform



Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Kuchen
kühlt aus

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Kuchen
kühlt aus

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

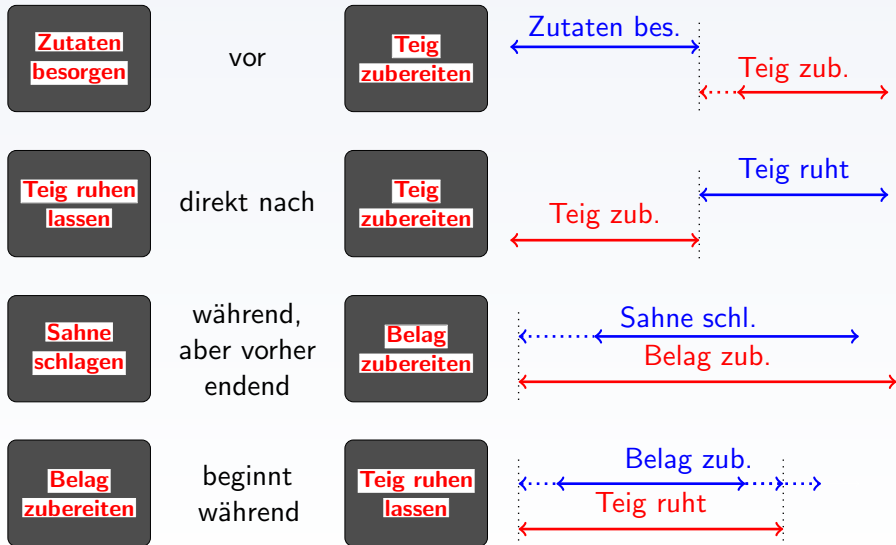
Kuchen im
Ofen

Kuchen
kühlt aus

Kuchen
entnehmen

Zutaten besorgen	Teig zubereiten	Teig ruhen lassen	Belag zubereiten	Sahne schlagen	Backform einfetten
Teig in Backform	Belag in Backform	Ofen heizt	Kuchen im Ofen	Kuchen kühlt aus	Kuchen entnehmen

- Aktionen entsprechen (nicht-leeren) **Zeitintervallen**
- Wissen: Anforderungen an die **relative** Lage der Intervalle
sonst nichts!: keine Zeit-Dauern
- Wie kann man dieses Wissen **repräsentieren**?
- Und wie daraus **Schlüsse 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?

- Neue Beziehungen zwischen Aktionen

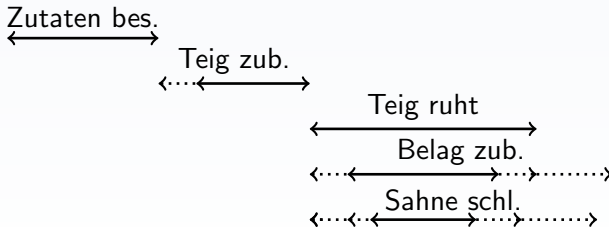
Darf der Belag vor dem Teig in die Form?

- 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

Communications 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 Zeit-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

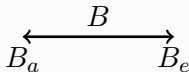
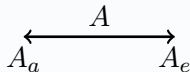
Allensche Formeln:

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

wobei

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

Basisrelationen: Gegeben zwei nichtleere reellwertige Intervalle:



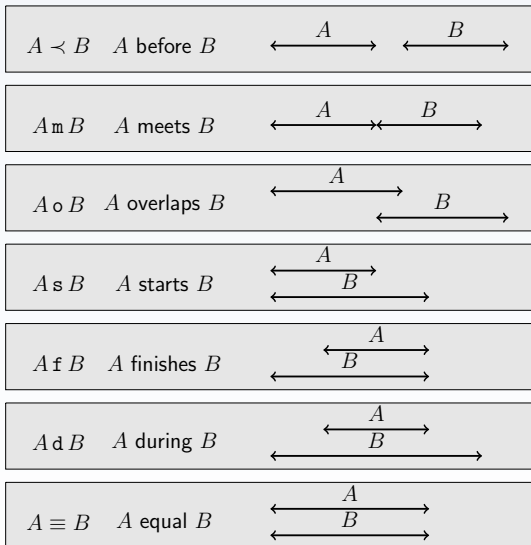
- Wie können A und B zueinander liegen?
- Wieviele Möglichkeiten gibt es?

Für $[A_a, A_e]$ und $[B_a, B_e]$ und $A_e \leq B_e$.

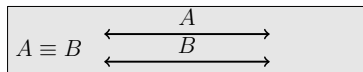
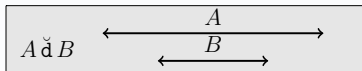
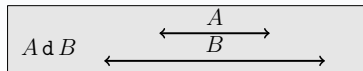
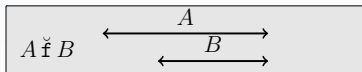
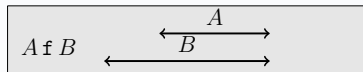
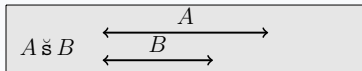
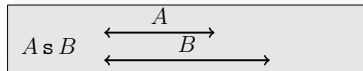
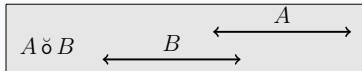
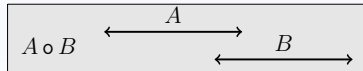
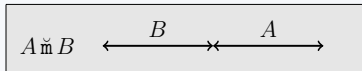
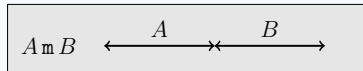
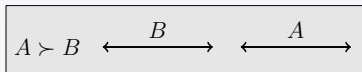
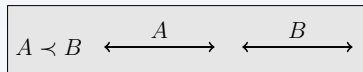
Andere Fälle mit $A_e \geq B_e$: A, B tauschen.

<i>Bedingung</i>	<i>Abkürzung</i>	<i>Bezeichnung</i>
$A_e < B_a$	$\prec$	A before B
$A_e = B_a$	m	A meets B
$A_a < B_a < A_e < B_e$	o	A overlaps B
$A_a = B_a < A_e < B_e$	s	A starts B
$B_a < A_a < A_e = B_e$	f	A finishes B
$B_a < A_a < A_e < B_e$	d	A during B
$B_a = A_a, A_e = B_e$	$\equiv$	A equal B

- und inverse Relationen (ohne $\equiv$)
- Inverse: $\check{r}$ ist inverse Relation zu r
- Ausnahmen:
 - $\succ$ inverse zu $\prec$
 - und $\equiv = \equiv$



Alle Allensche Basisrelationen



Allensche Basisrelationen

Die 13 Allenschen Basis-Relationen sind:

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

Fakt

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

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

Schreibweise zu Disjunktionen zu A, B

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

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

Dh. $2^{13} = 8192$ verschiedene atomare Allen-Constraints zu A, B .

Zutaten
besorgen

vor

Teig
zubereiten

Zutaten $\{\prec, m\}$ TeigZub

Teig ruhen
lassen

direkt nach

Teig
zubereiten

TeigR $\{\ddot{m}\}$ TeigZub

Sahne
schlagen

während,
aber vorher
endend

Belag
zubereiten

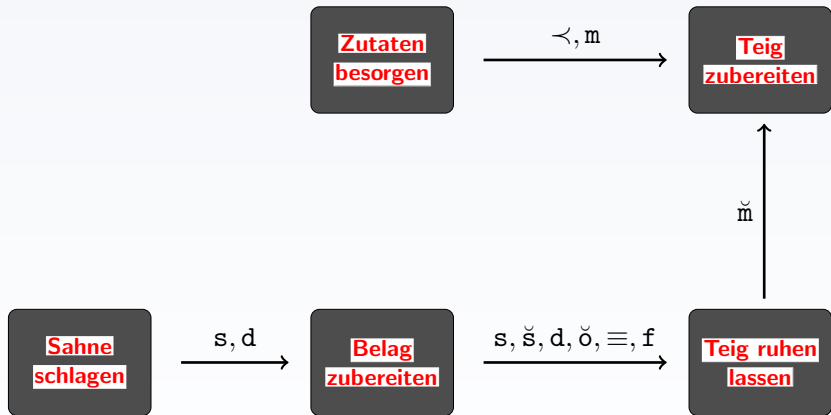
Sahne $\{s, d\}$ BelagZub

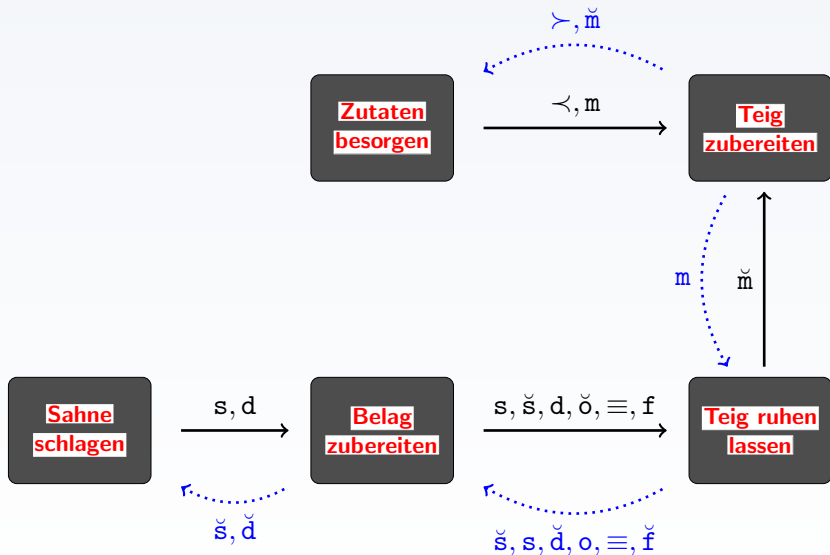
Belag
zubereiten

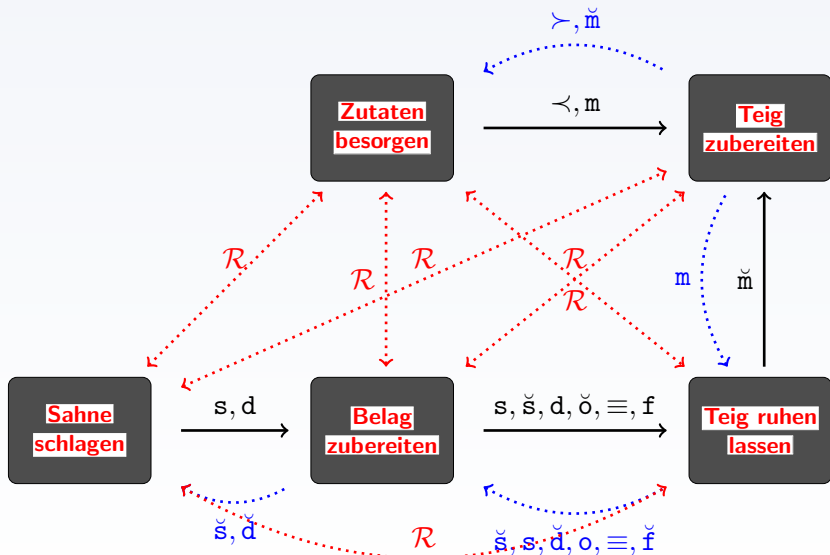
beginnt
während

Teig ruhen
lassen

BelagZub $\{s, \ddot{s}, d, \ddot{o}, \equiv, f\}$ TeigR







Interpretation I :

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

Interpretation von atomaren Aussagen $A \text{ r } B$:

Sei $I(A) = [A_a, A_e]$ und $I(B) = [B_a, B_e]$.

- $I(A \prec B) = 1$, gdw. $A_e < B_a$
- $I(A \text{ m } B) = 1$, gdw. $A_e = B_a$
- $I(A \circ B) = 1$, gdw. $A_a < B_a$, $B_a < A_e$ und $A_e < B_e$
- $I(A \text{ s } B) = 1$, gdw. $A_a = B_a$ und $A_e < B_e$
- $I(A \text{ f } B) = 1$, gdw. $A_a > B_a$ und $A_e = B_e$
- $I(A \text{ d } B) = 1$, gdw. $A_a > B_a$ und $A_e < B_e$
- $I(A \equiv B) = 1$, gdw. $A_a = B_a$ und $A_e = B_e$
- $I(A \check{r}_0 B) = 1$, gdw. $I(B \text{ r}_0 A) = 1$
- $I(A \succ B) = 1$, gdw. $I(B \prec A) = 1$

Interpretation I :

Für die Intervallenden reicht auch:

$\mathbb{Q}$ oder $\mathbb{Z}$ oder $\mathbb{N}$.

Aber:

Lässt man Intervalle der Länge 0 zu,
ergeben sich durch Sonderfälle weniger gute Transitivitätsregeln.

D.h. die Allen-Matrix (s.u.) ändert sich:

Z.B. gilt dann **nicht** mehr:

dass $A \sqsubseteq B$ und $B \sqsubseteq C$ die Relation $A \prec C$ impliziert.

Denn A, B, C könnten Länge 0 haben.

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. $I(F) = 1$ 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, $I(A \ \mathcal{R} \ B) = 1$.

- Ein atomare Aussage der Form $A \text{ r } A$ kann man immer vereinfachen zu 0, 1:
 - $A \text{ r } A \rightarrow 0$, wenn $r \neq \equiv$ und
 - $A \equiv A \rightarrow 1$.
- Negationszeichen kann man nach innen schieben.
- Eine Formel $\neg(A \text{ R } B)$ kann man zu $A (\mathcal{R} \setminus R) B$ umformen.
- Unterformeln der Form $A \text{ R}_1 B \wedge A \text{ R}_2 B$ kann man durch $A (\text{R}_1 \cap \text{R}_2) B$ ersetzen.
- Unterformeln der Form $A \text{ R}_1 B \vee A \text{ R}_2 B$ kann man durch $A (\text{R}_1 \cup \text{R}_2) B$ 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

① (konjunktives) Allen-Constraint:

Eine Konjunktion von atomaren Allen-Constraints:

$$A_1 \ S_1 \ B_1 \wedge \dots \wedge A_n \ S_n \ B_n$$

② Disjunktives Allen-Constraint:

Disjunktion von (konjunktiven) Allen-Constraints

Weniger geht nicht: Z.B. nicht vereinfachbar: $A \preceq B \vee C \preceq D$

- Eingabe: Allen-Constraint
- Ausgabe: Weitere Beziehungen die daraus folgen, bzw. 0 (Widerspruch) oder 1 (Tautologie)
- Es reichen im Grunde: Konjunktive Allen-Constraints
- Bei disjunktiven Allen-Constraints:
bearbeite die konjunktiven 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: $A R_1 B \wedge B R_2 C \models A(R_1 \circ R_2)C$.

	λ	γ	d	d̂	o	ō	≡	≡̂	s	ʒ	f	ĥ
λ	λ	ℛ	λ o ≡ d s	λ	λ	λ o ≡ d s	λ	λ o ≡ d s	λ	λ	λ o ≡ d s	λ
γ	ℛ	γ	γ ō ≡̂ d f	γ	γ ō ≡̂ d f	γ	γ ō ≡̂ d f	γ	γ ō ≡̂ d f	γ	γ	γ
d	λ	γ	d	ℛ	λ o ≡ d s	γ ō ≡̂ d f	λ	γ	d	γ ō ≡̂ d f	d	λ o ≡ d s
d̂	λ o ≡̂ d̂ ĥ ≡̂	γ ō ≡̂ d̂ ŝ ≡̂	ℛ / ≡̂ ≡̂	d̂	o d̂ ĥ	ō d̂ ŝ	o d̂ ĥ	ō d̂ ŝ	o d̂ ĥ	d̂	ō d̂ ŝ	d̂
o	λ	γ ō ≡̂ d̂ ŝ ≡̂	o d s	λ o ≡̂ d̂ f̂	λ o ≡̂	ℛ / ≡̂ ≡̂	λ	ō d̂ ŝ	o	d̂ ĥ o	d s o	λ o ≡̂
ō	λ o ≡̂ ĥ ≡̂	γ	ō d f	γ ō ≡̂ d̂ ŝ ≡̂	ℛ / ≡̂ ≡̂	γ ō ≡̂	o d̂ ĥ	γ	ō d f	γ ō ≡̂	ō	ō d̂ ŝ
≡	λ	γ ō ≡̂ d̂ ŝ ≡̂	o d s	λ	λ	o d s	λ	≡̂ ĥ ĥ	≡̂	≡̂	d s o	λ
≡̂	λ o ≡̂ ĥ ≡̂	γ	ō d f	γ	ō d f	γ	≡̂ ŝ ŝ	γ	d̂ ĥ o	γ	≡̂	≡̂
s	λ	γ	d	λ o ≡̂ d̂ ĥ ≡̂	λ o ≡̂	ō d f	λ	≡̂	s	≡̂ ŝ ŝ	d	λ o ≡̂
ʒ	λ o ≡̂ d̂ ĥ ≡̂	γ	ō d f	d̂	o d̂ ĥ	ō	o d̂ ĥ	≡̂	≡̂ ŝ ŝ	ʒ	ō	d̂
f	λ	γ	d	γ ō ≡̂ d̂ ŝ ≡̂	o d s	γ ō ≡̂	≡̂	γ	d	γ ō ≡̂	f	≡̂ ĥ ĥ
ĥ	λ	γ ō ≡̂ d̂ ŝ ≡̂	o d s	d̂	o	ō d̂ ŝ	≡̂	ō d̂ ŝ	o	d̂	≡̂ ĥ ĥ	ĥ

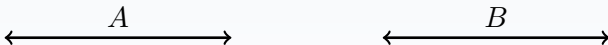
12 × 12-Matrix reicht, da:
 $r o \equiv \quad = \quad r \quad = \quad \equiv o r$

Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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

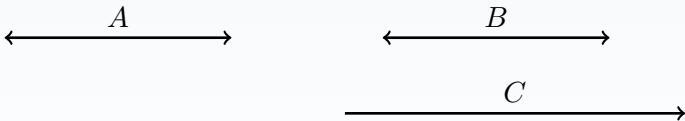


Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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

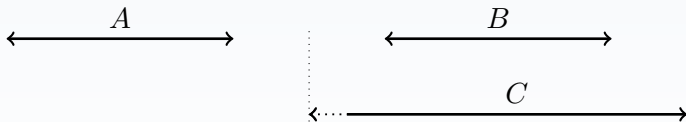


Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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

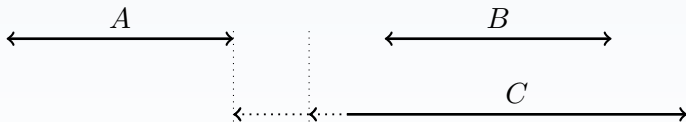


Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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



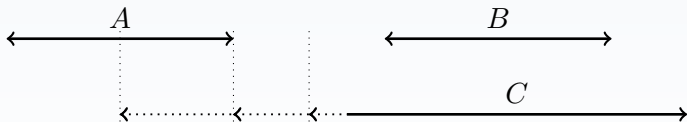
Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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



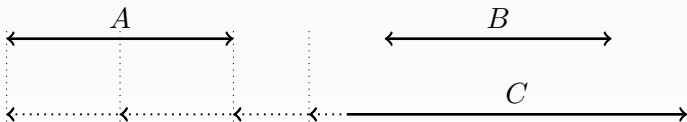
Möglichkeiten: $A \{ \prec, o, m, s, d \} C$.

Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

Beispiel: $\prec \circ d$

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



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

Die Einträge kann man per Hand ausrechnen.

Oder einmalige automatische Berechnung.

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 $S = \{r_1, \dots, r_k\} \subseteq \mathcal{R}$ und $\check{S} = \{\check{r}_1, \dots, \check{r}_k\}$.

Beachte. Es gilt: $\check{\check{r}} = r$

Damit gilt:

Satz

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

Satz

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

$$\overbrace{(r_1 \circ r_2)}^{\check{\cdot}} = \check{r}_2 \circ \check{r}_1$$

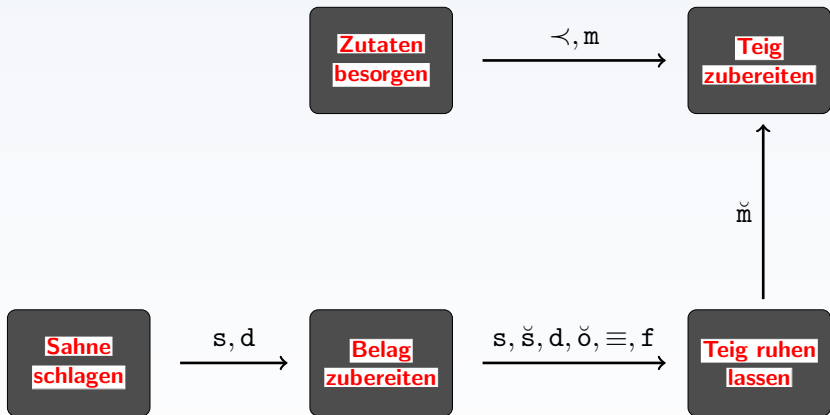
Eingabe: Konjunktives Allen-Constraint

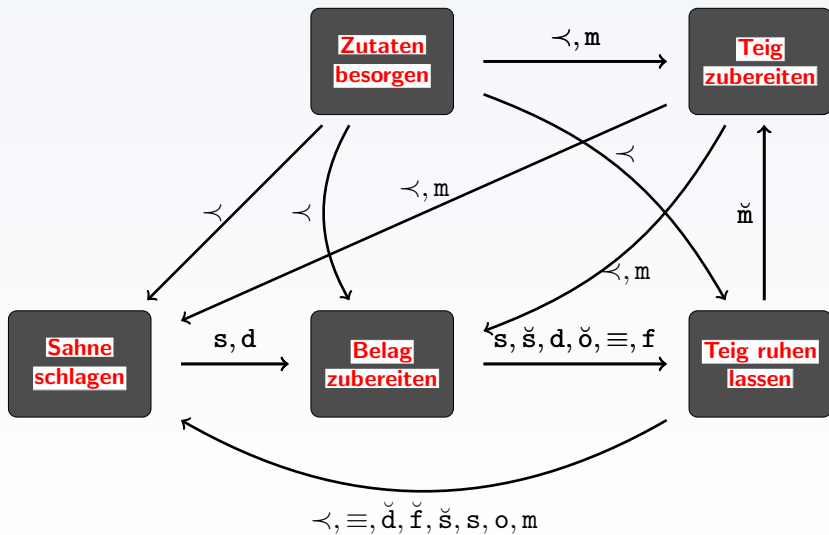
Ausgabe: Allenscher Abschluss

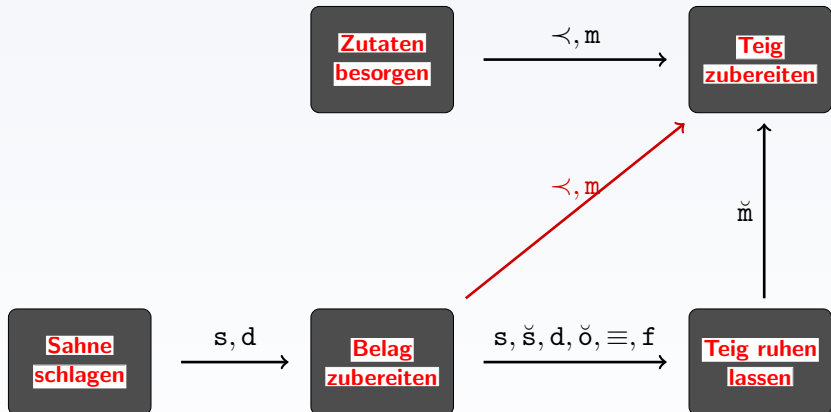
Verfahren: Berechne Fixpunkt bezüglich der Regeln (auf Subformeln):

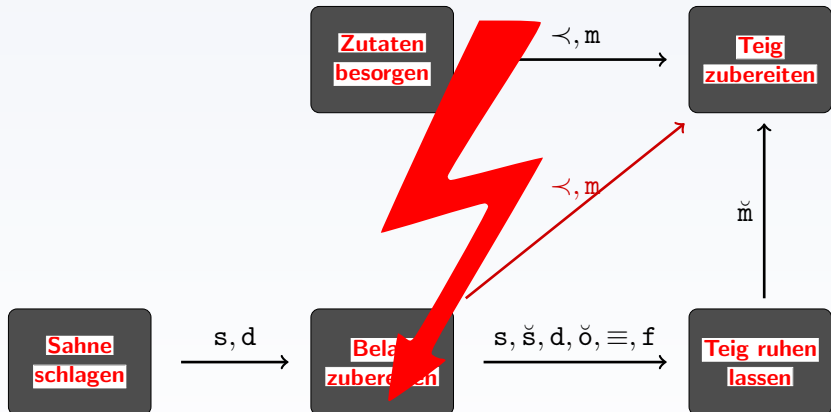
- Vereinfachungen: ($\rightarrow$ bedeutet „ersetze“)
 - $A R_1 B \wedge A R_2 B \rightarrow A (R_1 \cap R_2) B$
 - $A \emptyset B \rightarrow \text{False}$
 - $A \mathcal{R} B \rightarrow 1$
 - $A R A \rightarrow 0$, wenn $\equiv \notin R$.
 - $A R A \rightarrow 1$, wenn $\equiv \in R$.
- Folgerungen: ($\rightsquigarrow$ 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 (Disjunktion von konjunktiven Allensche Constraints):
Wende Fixpunktiteration auf jede Komponente an, und vereinfache anschließend
 - Komponente = 1: Disjunktiver Constraint ist äquivalent zu 1
 - Komponente = 0: Kann gestrichen werden
 - Alle Komponenten = 0: Disjunktiver Constraint widersprüchlich (Inkonsistenz)









Schlägt fehl, da der Schnitt an der roten Kante eine leere Kantenbeschriftung erzeugt.

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 unerfüllbare konjunktive Constraint herausfinden kann, dass 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?
- Was kann man über andere Zeitkalküle sagen?

- 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?
- Was kann man über andere Zeitkalküle sagen?

- 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 Relationen 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.

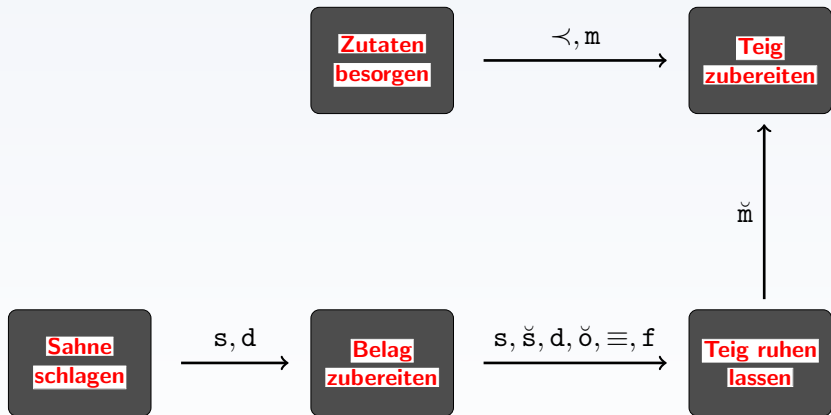
$$\begin{array}{c}
 \text{Zutaten } \{\prec, m\} \text{ TeigZub} \\
 \wedge \\
 \text{TeigRuht } \{\ddot{m}\} \text{ TeigZub} \\
 \wedge \\
 \text{Sahne } \{s, d\} \text{ Belagzub} \\
 \wedge \\
 \text{Belagzub } \{\equiv, s, \ddot{s}, d, f, \ddot{o}\} \text{ TeigRuht}
 \end{array}$$

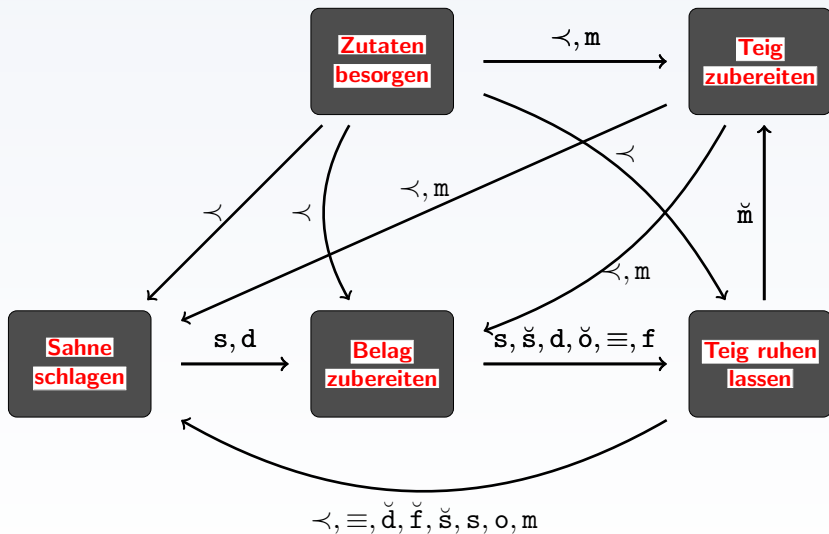
$R_{i,j}$	(1) Zutaten	(2) Teigzub	(3) TeigRuht	(4) Sahne	(5) Belagzub
(1) Zutaten	$\{\equiv\}$	$\{\prec, m\}$	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{R}$
(2) Teigzub	$\{\succ, \ddot{m}\}$	$\{\equiv\}$	$\{m\}$	$\mathcal{R}$	$\mathcal{R}$
(3) TeigRuht	$\mathcal{R}$	$\{\ddot{m}\}$	$\{\equiv\}$	$\mathcal{R}$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

$$\begin{array}{c}
 \text{Zutaten } \{\prec, m\} \text{ TeigZub} \\
 \wedge \\
 \text{TeigRuht } \{\check{m}\} \text{ TeigZub} \\
 \wedge \\
 \text{Sahne } \{s, d\} \text{ Belagzub} \\
 \wedge \\
 \text{Belagzub } \{\equiv, s, \check{s}, d, f, \check{o}\} \text{ TeigRuht}
 \end{array}$$

Vervollständigung:

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





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 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;

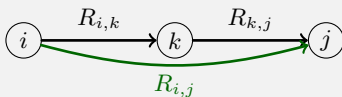
endif

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

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

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

entspricht gerade

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

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

entspricht gerade

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

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

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

Begründung:

- 3 geschachtelte for-Schleifen: $O(n^3)$
- repeat-Schleife: Im schlechtesten Fall wird nur 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;

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

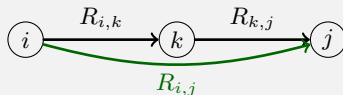
if $R_{i,j} \neq R'$ **then**

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

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

endif

endwhile



Algorithmus 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;

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

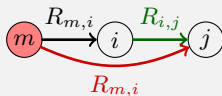
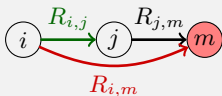
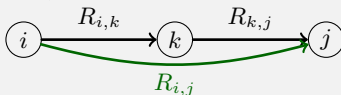
if $R_{i,j} \neq R'$ **then**

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

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

endif

endwhile



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

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

Korrektheit: Bei Änderung von $R_{i,j}$ werden alle Nachbarn, die evtl. neu berechnet werden müssen, in eine 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 $R_1 = \{r_1, \dots, r_k\}$, $R_2 = \{r'_1, \dots, r'_{k'}\}$.

$$\begin{aligned} & A R_1 B \wedge A R_2 B \\ = & (\bigvee A r_i B) \wedge (\bigvee A r'_{i'} B) \\ \sim & \bigvee \{(A r_i B) \wedge (A r'_{i'} B) \mid 1 \leq i \leq k, 1 \leq i' \leq k'\} \text{ (ausmultiplizieren)} \\ \sim & \bigvee \{(A r_i B) \wedge (A r'_{i'} B) \mid 1 \leq i \leq k, 1 \leq i' \leq k', r_i = r'_{i'}\} \text{ (Basisrelationen disjunkt)} \\ = & A (R_1 \cap R_2) B \\ & - A \emptyset B \sim 0 \text{ und } A \mathcal{R} B \sim 1 \text{ (klar)} \end{aligned}$$

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.

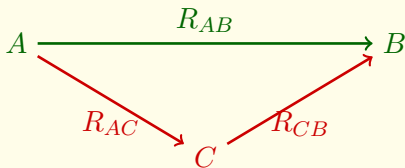
Für mehrelementige Mengen:

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

Der Allensche Kalkül ist vollständig in eingeschränktem Sinn:

Satz (Pfadkonsistenz)

Der Allensche Abschluss ist 3-konsistent:



D.h.: 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 $I(A \ R_{AC} \ C) = \text{True} = I(C \ R_{CB} \ B)$.

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:

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

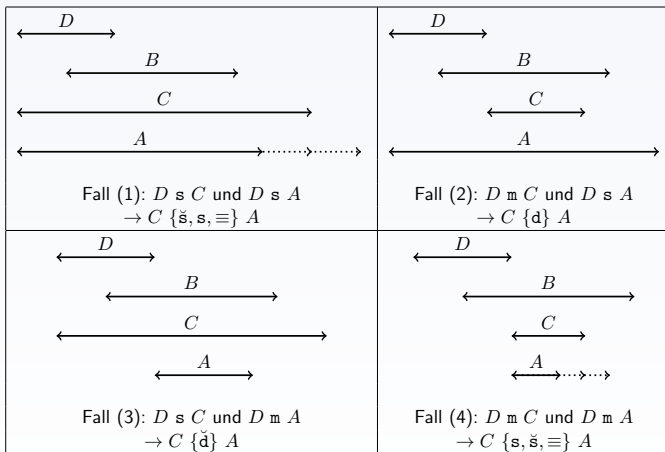
ist der Allensche Abschluss:

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

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

D.h. das Allen-Verfahren erkennt diese Unmöglichkeit nicht

- Die Lage von B zu D ist eindeutig.
- Möglichkeiten D zu A und D zu C : 4 Fälle $\{s, m\} \times \{s, m\}$



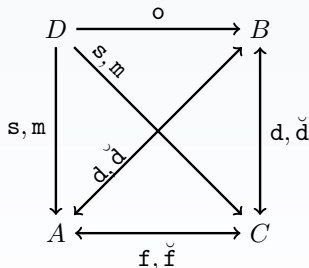
$C \{f, \check{f}, o, \check{o}\} A$ nicht möglich!

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 keinen 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

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):

$R' := \text{AllenAbschluss}(R)$;

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 $\mathcal{NP}$:

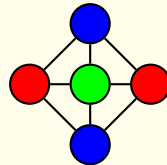
- Rate lineare Reihenfolge der Intervallanfänge und -enden
- D.h. Ordnung auf allen X_a, X_e für alle Intervalle X
- Verifiziere ob Reihenfolge das Constraint erfüllt
- Verifikation geht in Polynomialzeit

$\mathcal{NP}$ -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?



Für $G = (V, E)$ erzeuge:

Für $G = (V, E)$ erzeuge:

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

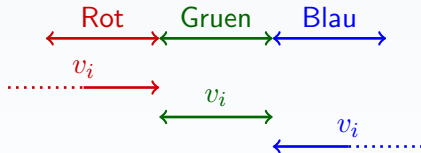


Für $G = (V, E)$ erzeuge:

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



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

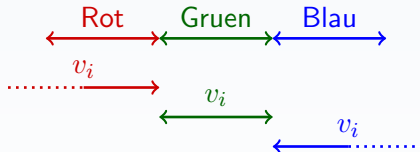


Für $G = (V, E)$ erzeuge:

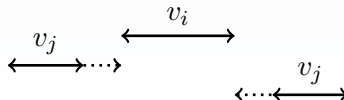
- $(\text{Rot} \text{ m Gruen}) \wedge (\text{Gruen} \text{ m Blau})$



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



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



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. $v_i \models \text{Gruen}$
- v_i hat Farbe blau gdw. $v_i \not\models \text{Gruen}$

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

- Jeder vollständige Algorithmus braucht Exponentialzeit.
(unter Annahme $\mathcal{NP} = EXPTIME$)
- Die polynomielle Allen-Vervollständigung ist **im allgemeinen unvollständig**

Es gibt polynomielle, vollständige Verfahren für

Allensche Constraints mit **eingeschränkter Syntax**

Eine haben wir bereits gesehen:

- Eindeutige Allen-Constraints

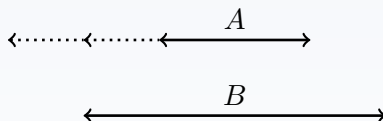
Neue Variante:

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

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



Wenn $A = [A_a, A_e]$, $B = [B_a, B_e]$, dann entspricht obige Relation gerade

$$A_a < A_e, B_a < B_e, A_e < B_e, B_a < A_e$$

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:

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

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

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

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

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

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))

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Kuchen
kühlt aus

Kuchen
entnehmen

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

Zutaten
besorgen

Teig
zubereiten

Teig ruhen
lassen

Belag
zubereiten

Sahne
schlagen

Backform
einfetten

Teig in
Backform

Belag in
Backform

Ofen
heizt

Kuchen im
Ofen

Kuchen
kühlt aus

Kuchen
entnehmen

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

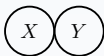
Anzahl Modelle : 1.536
Allenscher Abschluss genau? : True

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“