

Vorlesung Diskrete Strukturen

Ordnungsrelationen

Bernhard Ganter

Institut für Algebra
TU Dresden
D-01062 Dresden
bernhard.ganter@tu-dresden.de

WS 2009/10

Ein Spiegelei zubereiten . . .

Was muss man tun, wenn man ein Spiegelei brät?

- HE Herd einschalten
- PH Pfanne auf den Herd stellen
- PE Pfanne heiß werden lassen
- PF Fett in die Pfanne geben
- FZ Fett zerlassen
- EA Ei aufschlagen
- EP Ei in die Pfanne geben
- EB Ei gar braten
- ET Ei aus der Pfanne nehmen
- ES Ei salzen
- HA Herd abschalten

Precedence Constraints

Auf der Menge $J := \{\text{HE}, \text{PH}, \dots, \text{PE}\}$ definieren wir eine Relation R mit folgender Bedeutung:

$i R j : \iff$ Bevor Schritt j begonnen wird, muss Schritt i beendet sein.

durch

$$R := \{(\text{HE}, \text{PE}), (\text{PE}, \text{FZ}), (\text{FZ}, \text{EP}), (\text{EP}, \text{EB}), (\text{EB}, \text{ET}), (\text{EB}, \text{HA}), (\text{PH}, \text{PE}), (\text{PF}, \text{FZ}), (\text{EA}, \text{EP}), (\text{EA}, \text{ES})\}$$

Eine Eigenschaft solcher Relationen erkennt man sofort: sie sind **azyklisch**, d.h. sie enthalten keine Folgen $(a_0, a_1), (a_1, a_2), \dots, (a_{n-1}, a_n)$ mit $a_0 = a_n$. Insbesondere sind sie irreflexiv. Man spricht auch von einem **DAG** (directed acyclic graph).

Eine Eigenschaft solcher Relationen erkennt man sofort: sie sind **azyklisch**, d.h. sie enthalten keine Folgen $(a_0, a_1), (a_1, a_2), \dots, (a_{n-1}, a_n)$ mit $a_0 = a_n$. Insbesondere sind sie irreflexiv. Man spricht auch von einem **DAG** (directed acyclic graph).

Es lohnt sich aber nicht, eine eigene Theorie für DAGs aufzubauen, weil sie mit den Ordnungsrelationen ganz eng verwandt sind.

Eine Eigenschaft solcher Relationen erkennt man sofort: sie sind **azyklisch**, d.h. sie enthalten keine Folgen

$(a_0, a_1), (a_1, a_2), \dots, (a_{n-1}, a_n)$ mit $a_0 = a_n$. Insbesondere sind sie irreflexiv. Man spricht auch von einem **DAG** (directed acyclic graph).

Es lohnt sich aber nicht, eine eigene Theorie für DAGs aufzubauen, weil sie mit den Ordnungsrelationen ganz eng verwandt sind.

Eine **Ordnung** (oder Ordnungsrelation) auf einer Menge J ist eine *reflexive, transitive* und *antisymmetrische* Relation auf J .

Transitive Hülle

Die **transitive Hülle** einer Relation R ist definiert durch

$$\text{trans}(R) := \bigcup_{n \geq 1} R^n = R \cup R^2 \cup R^3 \cup \dots,$$

wobei R^n eine Abkürzung für das n -fache Relationenprodukt von R mit sich selbst steht, also

$$R^n := \underbrace{R; R; \dots; R}_{n\text{-mal}}.$$

Die transitive Hülle einer Relation R ist die kleinste transitive Relation, welche R enthält.

Hilfssatz

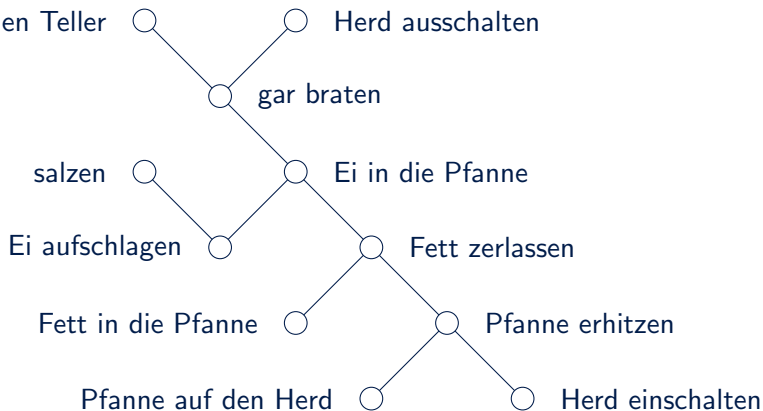
Eine Relation ist genau dann azyklisch, wenn ihre transitive Hülle irreflexiv ist.

Daraus folgt: Ist R eine azyklische Relation auf J , dann ist die reflexiv-transitive Hülle

$$\text{trans}(R) \cup \text{id}_J$$

eine Ordnungsrelation. Es ist oft einfacher, diese Ordnungsrelation zu betrachten.

Die Spiegelei-Ordnung



- Eine Relation auf einer Menge J wird eine *Ordnung* oder *Ordnungsrelation* genannt, wenn sie reflexiv, transitiv und antisymmetrisch ist.
- Andere Namen sind *Halbordnung* oder *Partialordnung*.
- Ist R eine Ordnung auf J , dann nennt man (J, R) eine *geordnete Menge*.
- Die englischen Bezeichnungen sind
 - *order*, *partial order* für „Ordnung“ und
 - *partially ordered set* für „geordnete Menge“, Kurzform: *poset*.
- Eine konnexe Ordnung nennt man auch eine *lineare Ordnung*.

Ordnungserweiterungen

Eine **Ordnungserweiterung** einer Relation R_1 auf J ist eine Ordnung R_2 auf J mit

$$R_1 \subseteq R_2.$$

Eine **Ordnungserweiterung** einer Relation R_1 auf J ist eine Ordnung R_2 auf J mit

$$R_1 \subseteq R_2.$$

Ist dabei R_2 eine lineare (d.h. konnexe) Ordnung, so spricht man von einer **linearen Erweiterung**.

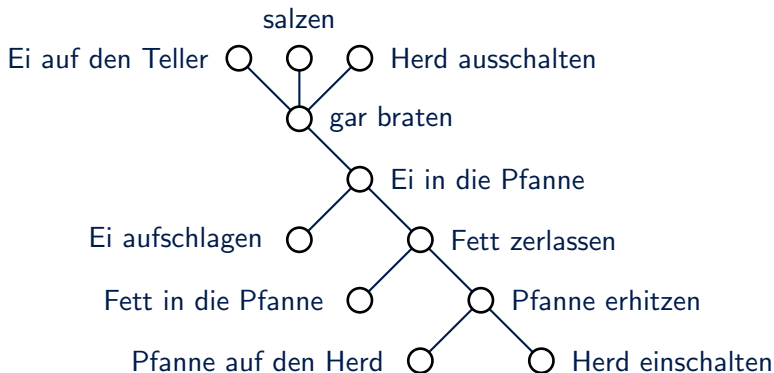
Eine **Ordnungserweiterung** einer Relation R_1 auf J ist eine Ordnung R_2 auf J mit

$$R_1 \subseteq R_2.$$

Ist dabei R_2 eine lineare (d.h. konnexe) Ordnung, so spricht man von einer **linearen Erweiterung**.

Die linearen Erweiterungen einer Precedence-Constraints-Relation entsprechen also genau den zulässigen Abläufen.

Wenn erst nach dem Braten gesalzen werden darf.



Das Lemma von Szpilrajn

Ein grundlegendes Ergebnis der Ordnungstheorie, das *Lemma von Szpilrajn*, besagt, dass eine Relation R genau dann eine Ordnungserweiterung besitzt, wenn $R \setminus \Delta_J$ azyklisch ist.

Das Lemma von Szpilrajn

Ein grundlegendes Ergebnis der Ordnungstheorie, das *Lemma von Szpilrajn*, besagt, dass eine Relation R genau dann eine Ordnungserweiterung besitzt, wenn $R \setminus \Delta_J$ azyklisch ist.

In diesem Fall besitzt sie sogar eine lineare Erweiterung.

Das Lemma von Szpilrajn

Ein grundlegendes Ergebnis der Ordnungstheorie, das *Lemma von Szpilrajn*, besagt, dass eine Relation R genau dann eine Ordnungserweiterung besitzt, wenn $R \setminus \Delta_J$ azyklisch ist.

In diesem Fall besitzt sie sogar eine lineare Erweiterung.

Gegebene Vorgängerbedingungen (Precedence Constraints) lassen also genau dann einen Ablaufplan zu, wenn sie azyklisch sind.

Ein Problem aus der Ablaufplanung

Das Arbeitsgebiet der **Ablaufplanung** (engl.: **Scheduling**) beschäftigt sich damit, zu vorgegebenen Jobs mit Precedence Constraints und anderen Randbedingungen in zulässige und möglichst optimale Reihenfolgen zu bringen.

Ein Problem aus der Ablaufplanung

Das Arbeitsgebiet der **Ablaufplanung** (engl.: **Scheduling**) beschäftigt sich damit, zu vorgegebenen Jobs mit Precedence Constraints und anderen Randbedingungen in zulässige und möglichst optimale Reihenfolgen zu bringen.

Wir stellen dazu ein Beispiel vor und einen Algorithmus von Lawler. Das Problem, das dieser Algorithmus löst, ist leicht zu verstehen, ebenso der Algorithmus. Gar nicht so einfach ist der Beweis, dass diese Vorgehensweise wirklich zum optimalen Ergebnis führt.

Ein kleines Beispiel

Eine Person soll sechs Arbeitsschritte $\{A, B, C, D, E, F\}$ ausführen, von denen jeder eine Zeiteinheit benötigt.

Dabei sollen folgende Vorgängerbedingungen beachtet werden:

$$\{(A, B), (A, E), (C, A), (C, F), (D, F), (F, B)\}.$$

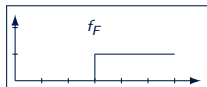
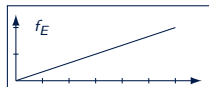
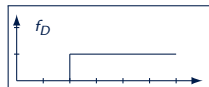
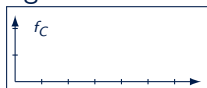
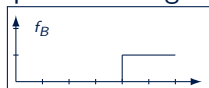
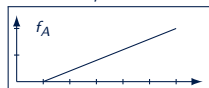
Ein kleines Beispiel

Eine Person soll sechs Arbeitsschritte $\{A, B, C, D, E, F\}$ ausführen, von denen jeder eine Zeiteinheit benötigt.

Dabei sollen folgende Vorgängerbedingungen beachtet werden:

$$\{(A, B), (A, E), (C, A), (C, F), (D, F), (F, B)\}.$$

Kosten, die bei verspäteter Fertigstellung anfallen:



Abstrakte Aufgabenstellung

- Betrachte eine endliche geordnete Menge (J, \leq) von Jobs, wobei wir die Ordnung wieder wie oben deuten: $i < j$ bedeutet, dass Job i abgeschlossen sein muss, bevor Job j begonnen werden kann.
- Alle Jobs werden von einer Person bearbeitet, Parallelverarbeitung ist also nicht vorgesehen.
- Jeder Job j benötigt eine *Ausführungszeit* p_j und soll ohne Unterbrechung durchgeführt werden.

Verzugskosten

Außerdem gibt es zu jedem Job auch *Verzugskosten*, die vom Fertigstellungszeitpunkt C_j abhängen:

$f_j(C_j)$ seien die Kosten, die anfallen, wenn der Job j zum Zeitpunkt C_j fertiggestellt wird. Dabei wird die Voraussetzung gemacht, dass die Funktionen f_j monoton in der Zeit sind: je später ein Job ausgeführt wird, desto teurer wird es.

Häufig wird der Spezialfall betrachtet, dass für jeden Job ein *Fälligkeitzeitpunkt* Z_j vorgegeben ist mit Verzugskosten

$$f_j(C) := \begin{cases} 0, & \text{falls } C \leq Z_j \\ 1, & \text{falls } C > Z_j \end{cases}.$$

Der Algorithmus von Lawler

Lawlers Algorithmus konstruiert einen Ablaufplan auf die folgende einfache Weise:

Lege zunächst fest, welcher Job als letzter ausgeführt wird. Wähle dazu unter allen Jobs, die bezüglich \leq keine Nachfolger haben, einen, dessen Verzugskosten für den betreffenden Zeitpunkt minimal sind. Für die verbleibenden Jobs verfare entsprechend.

Der Algorithmus von Lawler

Lawlers Algorithmus konstruiert einen Ablaufplan auf die folgende einfache Weise:

Lege zunächst fest, welcher Job als letzter ausgeführt wird. Wähle dazu unter allen Jobs, die bezüglich \leq keine Nachfolger haben, einen, dessen Verzugskosten für den betreffenden Zeitpunkt minimal sind. Für die verbleibenden Jobs verfare entsprechend.

Satz

Lawlers Algorithmus findet einen Ablaufplan, der erstens die gegebene Ordnung respektiert und zweitens die maximalen Verzugskosten

$$\max\{f_j(C_j) \mid j \in J\}$$

minimiert.

Beweis zum Algorithmus von Lawler

Beweis.

Für eine Teilmenge $N \subseteq J$ der Jobs bezeichne $f^*(N)$ die maximalen Verzugskosten für einen optimalen Ablaufplan von N . Weiter sei $q := \sum_{j \in J} p_j$ die Gesamtzeit und l ein Job ohne Nachfolger mit den kleinsten Verzugskosten zum Zeitpunkt q .

Die maximalen Verzugskosten für einen Ablaufplan mit l als letztem Job ergeben sich dann zu

$$\max\{f_l(q), f^*(J \setminus \{l\})\}.$$

Weil keiner der beiden Werte größer als $f^*(J)$ ist, ist die Wahl von l optimal. □

Modifiziertes Beispiel

Wie zuvor sind Arbeitsschritte $\{A, B, C, D, E, F\}$ auszuführen, von denen jeder eine Zeiteinheit benötigt, und für die folgende Vorgängerbedingungen beachtet werden sollen:

$$\{(A, B), (A, E), (C, A), (C, F), (D, F), (F, B)\}.$$

Modifiziertes Beispiel

Wie zuvor sind Arbeitsschritte $\{A, B, C, D, E, F\}$ auszuführen, von denen jeder eine Zeiteinheit benötigt, und für die folgende Vorgängerbedingungen beachtet werden sollen:

$$\{(A, B), (A, E), (C, A), (C, F), (D, F), (F, B)\}.$$

Die Verzugskosten seien nun:

