2

Reduktion

In diesem Kapitel studieren wir abstrakte Eigenschaften von Regeln. In den ersten beiden Abschnitten betrachten wir nicht einmal die Regeln selbst, sondern nur abstrakte Reduktionssysteme, die von Regeln erzeugt werden, und zeigen, wie sie benutzt werden können als Modell zur Berechnung von – im Allgemeinen mehrdeutigen – Funktionen, und als Mechanismus zur Definition von Sprachen. Erst in Abschnitt 2.3 betrachten wir, wie Regeln typischerweise Reduktionssysteme induzieren und damit Rechenmodelle oder Sprachen.

2.1 Abstrakte Reduktion

Dieser Abschnitt enthält nur den Standard-Stoff; siehe auch [Plu98, bschn. 1.2] und [BN98, Kap. 2].

Ersetzungssysteme (die manchmal auch Reduktionssystem genannt werden) definieren Berechnungen durch schrittweises Transformieren von Objekten. Die Objekte können Wörter, Terme, Formeln, Graphen oder andere Größen sein. Einige Konzepte und Eigenschaften von Ersetzungssystemen können unabhängig von der Art der transformierten Onjekte definiert und untersucht werden. So können abstrakte Eigenschaften von denen unterschieden werden, die von der Struktur der Objekte abhängen.

Abstrakte Reduktionssysteme sind Mengen mit einer binären Relation, die Transformationsschritte darstellt.

Definition 2.1 (Abstraktes Reduktionssystem). Ein *abstraktes Reduktionssystem* $\langle A, \to \rangle$ besteht aus einer Menge A und einer binären Relation $\to$ auf A.

Für den Rest dieses Kapitels soll $\langle A, \to \rangle$ ein beliebiges abstraktes Reduktionssystem bezeichnen. Für zwei Elemente a und b in A mit $\langle a, b \rangle \in \to$ schreiben wir $a \to b$. Die *Umkehrrelation* $\to^{-1} = \{\langle b, a \rangle \mid a \to b\}$ von $\to$ wird auch mit $\leftarrow$ bezeichnet, das *Komplement* $\not\to = (A \times A) \setminus \to$ enhält alle Paare, die nicht

in Relation stehen, und die *Komposition* zweier Relationen $\underset{1}{\to}$ und $\underset{2}{\to}$ auf A ist definiert als $\underset{1}{\to} \circ \underset{2}{\to} = \{\langle a, c\rangle \mid a \underset{1}{\to} b \text{ und } b \underset{2}{\to} c \text{ für ein } b \in A\}$.

Definition 2.2 (Abschlüsse).

1. Die *Identität* auf A ist die Relation $\overset{0}{\to} = \{\langle a, a\rangle \mid a \in A\}$.
2. Der *reflexive Abschluss* von $\to$ ist die Relation $\overset{=}{\to} = \to \cup \overset{0}{\to}$.
3. Jür jedes $n > 0$ ist die *n-fache Komposition* von $\to$ definiert als $\overset{n}{\to} = \to \cup \overset{n-1}{\to}$.
4. Der *transitive Abschluss* von $\to$ ist die Relation $\overset{+}{\to} = \bigcup_{n>0} \overset{n}{\to}$.
5. Der *transitiv-reflexive Abschluss* von $\to$ ist die Relation $\overset{*}{\to} = \overset{+}{\to} \cup \overset{0}{\to}$.
6. Der *symmetrische Abschluss* von $\to$ ist die Relation $\leftrightarrow = \leftarrow \cup \to$.
7. Der *Äquivalenz-Abschluss* von $\to$ (auch *Konvertierbarkeit* bzgl. $\to$ genannt) ist der transitiv-reflexive Abschluss von $\leftrightarrow$ und wird mit $\overset{*}{\leftrightarrow}$ bezeichnet.

Zwei Elemente a und b von A sind *ineinander konvertierbar* wenn $a \overset{*}{\leftrightarrow} b$ gilt. Ein Element a ist *in Normalform* wenn es kein Element b mit $a \to b$ gibt, und a *hat eine Normalform* wenn $a \overset{*}{\to} b$ für irgendeine Normalform b gilt; dann nennen wir b *eine Normalform von* a, sagen "a *reduziert* zu b" und schreiben $a \overset{!}{\to} b$. Ist b die einzige Normalform von a, bezeichnen wir b auch als $a\!\downarrow$.

Definition 2.3 (Termination und Konfluenz). Die Relation $\to$ ist

1. *endlich verzweigend* wenn für jedes Element a nur endlich viele Elemente b mit $a \to b$ existieren,
2. *terminierend* (anderswo auch *stark normalisierend* genannt) wenn es keine unendlichen Folgen $a_1 \to a_1 \to a_3 \to \cdots$ gibt,
3. *normalisierend* (anderswo auch *schwach normalisierend* genannt) wenn jedes Element aus A eine Normalform hat,
4. *eindeutig normalisierend* (bzw. $\to$ *hat eindeutige Normalformen*) wenn für alle Elemente a, b und c aus $b \overset{!}{\leftarrow} a \overset{!}{\to} c$ schon $b = c$ folgt.
5. *Church-Rosser* wenn es für alle Elemente a und b mit $a \overset{*}{\leftrightarrow} b$ ein Element c mit $a \overset{*}{\to} c \overset{*}{\leftarrow} b$ gibt,
6. *konfluent* wenn es für alle Elemente a, b und c mit $b \overset{*}{\leftarrow} a \overset{*}{\to} c$ ein Element d mit $a \overset{*}{\to} c \overset{*}{\leftarrow} b$ gibt,
7. *lokal* (oder *schwach*) *konfluent* wenn es für alle Elemente a, b und c mit $b \leftarrow a \to c$ ein Element d mit $a \overset{*}{\to} c \overset{*}{\leftarrow} b$ gibt,
8. *sub-kommutativ* wenn es für alle Elemente a, b und c mit $b \leftarrow a \to c$ ein Element d mit $a \overset{=}{\to} c \overset{=}{\leftarrow} b$ gibt,
9. *konvergent* wenn sie terminierend und konfluent ist.

Lemma 2.4 (Lemma 2.4 in [Plu99b]).

1. *Temination impliziert Normalisierung.*
2. *Die Church-Rosser-Eigenschaft ist äquivalent zu Konfluenz.*

3. Subkommutativität impliziert Konfluenz.
4. Konfluenz impliziert lokale Konfluenz.
5. Konfluenz impliziert eindeutige Normalformen.

Nach Aussage 2.4.5 hat in einer konfluenten Relation jedes Element höchstens eine Normalform. Die Umkehrungen der Aussagen 1, 3, 4, und 5 gelten nicht. Für Aussage 4 macht das Beispiel der lokal konfluenten (und normalisierenden) Relation $c \leftarrow a \leftrightarrows b \rightarrow d$ dies deutlich. Für terminierende Relationen sind lokale und allgemeine Konfluenz aber äquivalent.

Lemma 2.5 (Newman's Lemma). *Eine terminierende Relation ist genau dann konfluent wenn sie lokal konfluent ist.*

2.1.1 Abstrakte Reduktion als Rechenmodell

Ein abstraktes Reduktionssystem $\langle A, \rightarrow \rangle$ kann dazu benutzt werden, eine nichtdeterministische Funktion auf der Menge A zu definieren.

Eine mehrdeutige Funktion f von einer Menge A in eine Menge B wird als $f\colon A \multimap B$ notiert und ist eine "normale" Funktion in die Potenzmenge $\wp(B)$. eine mehrdeutige Funktion heißt *eindeutig* wenn $|f(a)| \leqslant 1$ für alle $a \in A$, bzw. *total* wenn $|f(a)| \geqslant 1$ für alle $a \in A$.[1]

Die vom abstrakten Reduktionssystem $\langle A, \rightarrow \rangle$ *induzierte mehrdeutige Funktion* $\vec{f}\colon A \multimap A$ ist definiert als

$$\forall a \in A : \vec{f}(a) = \{b \in A \mid a \xrightarrow{!} b\}$$

Dann gilt:

Lemma 2.6. *1. Die Funktion $\vec{f}$ ist eindeutig genau dann, wenn $\langle A, \rightarrow \rangle$ konfluent ist.*
2. Die Funktion $\vec{f}$ ist total genau dann, wenn $\langle A, \rightarrow \rangle$ normalisierend ist.

Deshalb sind Konfluenz und Normalisierung wichtige Eigenschaften, wenn wir Reduktionssysteme zur Berechnung von Funktionen einsetzen wollen.

2.1.2 Strategien

Als kleine Ergänzung zu den klassischen Begriffen der abstrakten Reduktion betrachten wir noch allgemeine Eigenschaften von Strategien. Mit Strategien der (hier noch nicht beschriebenen) Regelanwendung wählt man eine Teilrelation von $\rightarrow$ aus.

Definition 2.7. Eine Teilrelation $\xrightarrow{s} \subseteq \rightarrow$ nennen wir *strategische Einschränkung.*

Die strategische Einschränkung $\xrightarrow{s}$ ist

[1] Hierbei bezeichnet $|A|$ die *Kardinalität* der Menge A, also die Zahl ihre Elemente.

- *normalisierend* wenn aus $a \xrightarrow{!} b$ immer $a \xrightarrow{s}{}^{!} b$ folgt,
- *deterministisch*, wenn jedes $a \in A$ nur höchstens ein $b \in B$ mit $a \xrightarrow{s} b$ hat, und
- *optimal* wenn für alle Elemente a und b mit $a \xrightarrow{n} b$ und $a \xrightarrow{s}{}^{m} b$ gilt: $n \geqslant m$.

Deterministische normalisierende Strategien sind für die praktische Anwendung von Reduktionssystemen interessant. Optimale natürlich auch, nur sind sie schwieriger zu finden.

2.2 Abstrakte Sprachen

Ein abstraktes Reduktionssystem $\langle A, \rightarrow \rangle$ kann auch dazu benutzt werden, eine *Sprache* zu definieren. Abstrakt gesehen ist eine *Sprache* lediglich eine Teilmenge $L \subseteq A$ der Objekte des Reduktionssystems.

Typischerweise wird dazu zunächst eine Teilmenge $A_\mathrm{t} \subseteq A$ von *terminalen Elementen* von A charakterisiert, in der die Sprache L enthalten sein soll, und ein *Startelement* $z \in A$ ausgezeichnet.

So eine *abstrakte Grammatik* $\Gamma = \langle A, A_t, \rightarrow, z \rangle$, erzeugt die Sprache

$$L_\Gamma = \{a \in A_t \mid z \xrightarrow{*} a\} \subseteq A_t$$

Grammatiken (ob abstrakt oder konkret) definieren, wie die Wörter einer Sprache abgeleitet werden können. Praktisch ist man jedoch oft an der Antwort auf eine etwas andere Frage interessiert.

Definition 2.8 (Wortproblem). Das *Wortproblem* für eine Grammatik Γ lautet:

$$\text{Gilt } a \in L_\Gamma \text{ für irgendein } a \in A_\mathrm{t}?$$

Das Wortproblem ist für Γ *entscheidbar*, wenn es einen Algorithmus gibt, der diese Frage für alle Elemente von A entscheidet, d.h. *true* liefert genau dann $w \in L_\Gamma$, und *false* sonst.

Für eine Klasse $\boldsymbol{\Gamma}$ von Grammatiken ist das Wortproblem dann entscheidbar, wenn es für jede Grammatik $\Gamma \in \boldsymbol{\Gamma}$ entscheidbar ist.

2.3 Abstrakte Regeln

In abstrakten Reduktionssystemen $\langle A, \rightarrow \rangle$ wird von Regeln noch abstrahiert, sondern nur Mengen und Relationen (Ersetzungsschritte) darauf betrachtet. In allen interessanten Fällen sind die Mengen und Ersetzungsschritte *unendlich*.

Wenn wir Rechenschritte beschreiben wollen, so sind das letztendlich *Algorithmen*, und die müssen bekanntlich eine *endliche Beschreibung* haben. Und hier kommen die *Regeln* ins Spiel. Im Folgenden werden wir immer von einer

endlichen Regelmenge $R \subseteq \mathcal{R}$ aus einer (im Allgemeinen unendlichen) Menge $\mathcal{R}$ aller prinzipiell zu bildenden Regeln ausgehen, die eine (i. Allg. unendliche) Reduktionsrelation $\rightarrow_R$ *induziert*.

Wenn die Regeln benutzt werden sollen, aus einem Element $a \in A$ ein anderes Element $b \in B$ mit $a \rightarrow_R b$ zu berechnen, brauchen wir auch noch eine *operationale Semantik* der Regeln. Dafür definieren wir einen *Anwendungsoperator*, der beliebige Regeln auf ein Element von A anwenden kann und alle möglichen Reduktionsergebnisse liefert. Der Anwendungsoperator ist also eine mehrdeutige Funktion $@\colon \mathcal{R} \times A \multimap A$, für den gilt:

$$b \in (l/r)@a \text{ genau dann wenn } a \rightarrow_{\{l/r\}} b$$

Der Anwendungsoperator für Regeln soll berechenbar sein, und zwar möglichst effizient, weil er die Grundlage der prototypischen Systeme bildet, die aus einem Regelsystem generiert werden.

Literaturhinweise

Die Betrachtung von Eigenschaften abstrakter Reduktion ist üblich seit Huét [Hue80]. Wir finden das bei Avenhaus [Ave95], Baader und Nipkow [BN98], Klop [Klo92] und Plump [Plu99b]. Für Grammatiken kann man nicht so viele Eigenschaften abstrakt untersuchen.