Kapitel 8

Reguläre Relationen

8.1 Motivation

Wir haben gesehen, daß wir mit regulären Baumsprachen gut rechnen können, weil sie durch endliche Automaten dargestellt werden.

Jetzt interessieren wir uns nicht mehr für Sprachen, also Mengen von Bäumen, sondern Relationen, also Mengen von Paaren von Bäumen. Welche Relationen lassen sich effektiv handhaben?

Wir sind eigentlich interessiert an Relationen, die durch Term-Ersetzungs-System erzeugt werden, denn diese erhalten wir bei der Analyse von (funktionalen) Programmen.

Diese Klasse ist aber viel zu groß und kompliziert (Turing-vollständig), deswegen suchen wir sinnvolle Teilklassen. Eine wesentliche Forderung wird sein, daß die Klasse abgeschlossen ist gegen Bildung transitiver Hüllen.

Nach zwei „trivialen“ Ansätzen (der erste ist zu einfach, der zweite zu allgemein) beschreiben wir GTTs (Ground Tree Transducer), die manche schöne Eigenschaften besitzen.

8.2 Einfache Reguläre Relationen

8.2.1 (Wünschenswerte) Eigenschaften von Relationen

Für Relationen R, S und Sprachen L sind wir interessiert an

- Projektion $R(L) = \{y \mid x \in L, (x, y) \in R\}$
- Komposition $R \circ S$
- transitiver Hülle R^*

8.2.2 Kreuzprodukte von Sprachen

endliche Vereinigungen von Kreuzprodukten regulärer Mengen

abgeschlossen unter alle genannten Operationen

aber sehr einfache Klasse (Beispiel: Identität nicht enthalten)

8.2.3 Kreuzprodukte von Alphabeten

Durch Alphabetvergrößerung Paar (i. A. Tupel) von Bäumen in einem Baum kodieren.

reichhaltige Klasse (enthält Identität)

nicht abgeschlossen unter transitiver Hülle.

8.3 Ground Tree Transducer

8.3.1 Definition

(Dauchet, Tison, 198?)

Def. Ein GTT ist ein Paar von Automaten (A, B) über dem gleichen Alphabet mit (einigen) gemeinsamen Zuständen $Q_{A,B} := Q_A \cap Q_B$, diese heißen Synchron-Zustände.

Def. Ein GTT (A, B) akzeptiert die Relation R mit $(s, t) \in R \iff$

$$\begin{aligned}\exists C[\ldots], t_1, \ldots, t_n, s_1, \ldots, s_n, q_1, \ldots, q_n \in Q_{A,B} :\\ t = C[t_1, \ldots, t_n], s = C[s_1, \ldots, s_n],\\ \forall i : t_i \in \mathsf{Lang}(A, q_i) \wedge s_i \in \mathsf{Lang}(B, q_i)\end{aligned}$$

8.3.2 Eigenschaften

GTTs sind abgeschlossen unter Komposition und Iteration. Interessant ist die Iteration. Dafür gibt es einen Beweis (siehe TATA-Buch), der direkt die Konstruktion des GTT für R^* verifiziert.

Ein anderer Beweis (Engelfriet 199?) verwendet einen netten Trick: $R^*(x, y)$ bedeutet die Existenz einer Rechnung des GTT R von x nach y, über eventuell mehrere Zwischenschritte. Diese Berechnung kodieren wir als Baum. (Das ist der Trick!) Die Menge dieser Berechnungsbäume ist regulär. (Deswegen funktioniert er!)

Wir verdeutlichen das für den Fall eines Alphabets Σ mit ausschließlich unären Symbolen (die Bäume sind also Wörter): Wir legen den linken Wortanfang als Baum-Wurzel fest. Ein GTT darf, von rechts kommend, einen Suffix abschneiden und durch einen anderen ersetzen. An der Schnittstelle fügen wir ein neues binäres Symbol B ein, das als linkes Argument den alten und als rechtes den neuen Suffix erhält.

Die Menge aller möglichen Berechnungsbäume ist enthalten in $\mathsf{Tree}(\Sigma \cup B)$. Ein Berechnungsbaum ist korrekt, wenn jeder Pfad von einem Blatt nach oben und wieder herunter zum direkten Nachbarn des Blattes genau einer GTT-Anwendung entspricht.

Wir verlangen also, daß ein Berechnungsbaum keine unkorrekten Teilbaum enthält. Die korrekten Teilbäume stellen wir durch inverse Homomorphismen dar. Der eine Hom. ist $l(B(x, y)) = x$, der andere $r(B(x, y)) = y$.