Frank Morawietz[†] **MSO Logik, Linguistik und Baumautomaten** 05.05.97

Tag des Baumes: GK ILS

1 Einführung

Modell-Theoretische Syntax: Blackburn, Kracht, Rogers und Stabler (siehe Literaturliste oder Übersicht in Cornell (1996))

- Wohlgeformtheitsbedingungen charakterisieren die gewünschten Strukturen – in unserem Fall Bäume – vollständig deklarativ
- Monostratale Sicht auf Grammatikformalismen
- Constraint-basiert: Formale Sprache mit Syntax (MSO Logik) und Semantik (Bäume)[1]

2 Eine Formale Sprache: Monadische Prädikatenlogik zweiter Stufe auf Bäumen

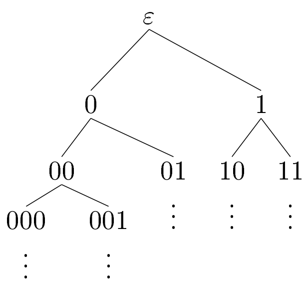
2.1 Modelle

Definition 2.1 *Eine **Baum-Domäne** T ist eine nicht leere Menge von Zeichenketten über einem linear geordneten Alphabet, die abgeschlossen ist unter Präfix (1) und linker Schwester (2).*
$T \subseteq \mathbb{N}^*$ ist eine Baum-Domäne, wenn für alle $u, v \in \mathbb{N}^*$ und für alle $i, j \in \mathbb{N}$ gilt:

1. *$uv \in T \Rightarrow u \in T$ und*
2. *$ui \in T$ und $j < i \Rightarrow uj \in T$.*

Die Intuition hier ist, daß wir Adressen in einem Baum betrachten. Die Wurzel hat die Adresse ε, während die Töchter eines Knotens u von links nach rechts die Adressen $u \cdot 0$, $u \cdot 1$, ... erhalten. Ein Beispiel über ein Alphabet aus 0 und 1, d.h. binäre Bäume:

(1)



Die n-verzweigende Baum-Domäne bezeichnen wir mir $T_n = \{i \mid i < n \in \mathbb{N}\}$, d.h. jeder Knoten hat n Töchter. Sei S_i die i-te Nachfolgerrelation, d.h. $S_i(u, v) \Leftrightarrow u = v \cdot i$ und $\leq$ die Präfixrelation.

Definition 2.2 *Die **Struktur von n-Nachfolger Funktionen** $\mathcal{N}_n$ ist definiert als*

$$\mathcal{N}_n = \langle T_n, \varepsilon, \leq, S_i, \rangle_{i<n}.$$

[†]Was hier auf dem Handout steht, ist natürlich nicht alles auf meinem Mist gewachsen. Die Hauptquelle ist Rogers (1994). Teile stellen auch Übersetzungen von gemeinsamer Arbeit mit Tom Cornell dar.

[1]Vorsicht hier, das ist nicht die Semantik für natürliche Sprache!

2.2 Die Sprache

Definition 2.3 *Eine MSO-Sprache $\mathcal{L}_{MSO}$ auf Bäumen besteht aus:*

- *Einer Menge* Monadischer *Variablen 2.Ordnung (X,Y, ...);*
- *Einer Menge von Individuenvariablen (x,y, ...);*
- *booleschen Junktoren:* $\neg$, $\wedge$*;*
- *Quantoren über beide Arten von Variablen:* $\exists$*;*
- *Binären Prädikate für die Nachfolgerrelationen:* s_i*;*
- *Präfix, Gleichheit und Element:* $\leq$, $\approx$, $\in$.

Bemerkungen: Da die Sprache über n-Nachfolgerfunktionen gebildet ist, spricht man oft von der Theorie der Nachfolgerfunktionen und bezeichnt sie mit SnS. Man muß zwischen *starken* und *schwachen* Versionen unterscheiden. In der starken können die Variablen 2.Stufe auch infinite Mengen repräsentieren. Die meisten Resultate sind im endlichen Fall jedoch viel leichter, und da in der Linguistik sowieso nur endliche Bäume betrachtet werden, werden wir uns im Normalfall auf endliche Mengen beschränken.
Wir machen freien Gebrauch von den gebräuchlichen "Abkürzungen", z.B. $\phi \vee \psi$, $\phi \Rightarrow \psi$, $\phi \Leftrightarrow \psi$, und $(\forall X)\phi$ für $\neg(\neg\phi \wedge \neg\psi)$, $\neg(\phi \wedge \neg\psi)$, $\neg(\phi \wedge \neg\psi) \wedge \neg(\neg\phi \wedge \psi)$, und $\neg(\exists X)\neg\phi$.

Definition 2.4 ***Terme*** *für $\mathcal{L}_{MSO}$ sind nur die Variablen.*[2] *Die* ***wohlgeformten Formeln*** *(WFF) von $\mathcal{L}_{MSO}$ sind als die kleinste Menge definiert, die die folgenden Bedingungen erfüllt:*

- *Wenn x, y Individuenvariablen sind und s_i ist eine Nachfolgerrelation, dann ist $s_i(x, y)$ eine WFF;*
- *Wenn x, y Individuenvariablen sind und X eine Mengenvariable ist, dann sind auch $x \approx y$, $x \leq y$ und $x \in X$ (alternativ manchmal auch einfach als $X(x)$ geschrieben) WFF;*
- *Wenn φ und ψ WFF sind, dann sind auch $\neg\varphi$, $\varphi \wedge \psi$ und $\varphi \vee \psi$ WFF;*
- *Wenn x, X Variablen sind und φ eine Formel ist, dann sind auch $(\exists x)[\varphi]$ bzw. $(\exists X)[\varphi]$ und $(\forall x)[\varphi]$ bzw. $(\forall X)[\varphi]$ WFF;*
- *Wenn P ein n-stelliges Prädikatensymbol ist und $t_1, \ldots, t_n$ sind Terme, dann ist $P(t_1, \ldots, t_n)$ eine WFF.*

Sätze *sind WFF ohne freie Variablen.*

2.3 Erfüllbarkeit

Definition 2.5 ***Erfüllbarkeit*** *ist (wie immer) relativ zu einer Variablenzuweisung.*

$$\mathcal{N}_n \models \phi(x_1, \ldots, x_n, X_1, \ldots, X_m)[\alpha]$$

wobei α Individuenvariablen zu Knoten (Elemente von T_n) und (monadische) Variablen 2.Stufe zu Mengen von Knoten (Teilmengen von T_n^i für geeigntes i) abbildet.

[2] Wenn die Sprache Konstanten hätte, wären diese auch Terme.

2.4 Definierbarkeit

Lemma 2.6 *Wir können die Signatur von $\mathcal{L}_{MSO}$ mit beliebigen Prädikaten erweitern, so lange die Relationen*

- *explizit definierbar (siehe Gleichungen (2) oder (3)) oder*
- *monadisch und induktiv-elementar definierbar sind.*

(2) $$X \subseteq Y \overset{def}{\Longleftrightarrow} (\forall x)[x \in X \Rightarrow x \in Y]$$

(3) $$\begin{aligned} \mathsf{Sing}(X) \overset{def}{\Longleftrightarrow} & \\ & \text{% Alle Teilmengen von } X \text{ sind gleich } X \text{ oder leer:} \\ & (\forall Y)[Y \subseteq X \Rightarrow (X \subseteq Y \vee (\forall Z)[Y \subseteq Z]\,)] \wedge \\ & \text{% } X \text{ ist nicht die leere Menge:} \\ & (\exists Y)[X \not\subseteq Y] \end{aligned}$$

2.5 Ein Beispiel: $\mathcal{L}^2_{K,P}$

Definition 2.7 *Sei $\mathcal{L}^2_{K,P}$ (Rogers 1994) definiert durch:*

- *(abzählbar unendliche) Mengen* **K** *und* **P** *von Individuen- bzw. Prädikatenkonstanten;*
- *eine (abzählbar unendliche) Menge* **X** *Monadischer Variablen 2.Ordnung;*
- *boolesche Junktoren: $\neg$, $\wedge$, $\vee$;*
- *Quantoren: $\exists$, $\forall$;*
- *Prädikate für*

$\triangleleft$	$--$	*direkte Dominanz*
$\triangleleft^*$	$--$	*reflexive Dominanz*
$\triangleleft^+$	$--$	*echte Dominanz*
$\prec$	$--$	*echte Präzedenz*
$\approx$	$--$	*Gleichheit*

- *Element: $\in$.*

$\mathcal{L}^2$ ist dieselbe Sprache ohne Konstanten.

Definition 2.8 *Ein **Modell** für $\mathcal{L}^2_{K,P}$ ist ein Tupel $\langle \mathcal{U}, \mathcal{I}, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mathcal{R}_p \rangle_{p \in \mathbf{P}}$ mit:*

$\mathcal{T}$	*eine Baum-Domäne*
$\mathcal{I}$	*eine Funktion von* **K** *nach $\mathcal{T}$*
$\mathcal{P}$, $\mathcal{D}$, $\mathcal{L}$	*binäre Relationen auf $\mathcal{T}$, die $\triangleleft$, $\triangleleft^*$ und $\prec$ interpretieren*
$\mathcal{R}_p$	*Relationen geeigneter Stelligkeit auf $\mathcal{T}$, die $p \in \mathbf{P}$ interpretieren.*

Ein Modell für $\mathcal{L}^2$ ist einfach ein Tupel $\langle \mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L} \rangle$.

Theorem 2.9 (Rogers (1994)) *$\mathcal{L}^2$ ist "äquivalent" zu $\mathcal{L}_{MSO}$.*

2.5.1 Eine Axiomatisierung

Definition 2.10 *Wir definieren $\mathcal{A}_T$ als die Menge der folgenden Axiome:*

Rootedness: $(\exists x)(\forall y)[x \lhd^* y]$

Antisymmetry ($\lhd^*$): $(\forall x, y)[(x \lhd^* y \wedge y \lhd^* x) \rightarrow x \approx y]$

Transitivity ($\lhd^*$): $(\forall x, y, z)[(x \lhd^* y \wedge y \lhd^* z) \rightarrow x \lhd^* z]$

Parent 1: $(\forall x, y, z)[x \lhd y \rightarrow (x \lhd^+ y \wedge ((x \lhd^* z \wedge z \lhd^* y) \rightarrow (z \lhd^* x \vee y \lhd^* z)))]$

Parent 2: $(\forall x, z)(\exists y)[z \lhd^+ x \rightarrow y \lhd x]$

Path: $(\forall x, z)(\exists y)[x \lhd^+ z \rightarrow (x \lhd y \wedge y \lhd^* z)]$

Exhaustiveness: $(\forall x, y)[x \lhd^* y \vee y \lhd^* x \vee x \prec y \vee y \prec x]$

Exclusiveness: $(\forall x, y)[x \prec y \rightarrow (\neg(x \lhd^* y) \wedge \neg(y \lhd^* x) \wedge \neg(y \prec x))]$

Non–Tangling: $(\forall w, x, y, z)[(x \prec y \wedge x \lhd^* w \wedge y \lhd^* z) \rightarrow w \prec z]$

Transitivity ($\prec$): $(\forall x, y, z)[(x \prec y \wedge y \prec z) \rightarrow x \prec z]$

Precedence: $(\forall w)[(\exists x)[w \lhd^+ x] \rightarrow (\exists y)[w \lhd y \wedge (\forall z)[w \lhd^+ z \rightarrow \neg(z \prec y)]]]$

Well foundedness ($\lhd^+$): $(\forall X)[(\exists x)[X(x)] \rightarrow (\exists y)[X(y) \wedge (\forall z)[z \lhd^+ y \rightarrow \neg X(z)]]]$

Well foundedness ($\prec$): $(\forall X)[(\exists x)[X(x)] \rightarrow (\exists y)[X(y) \wedge (\forall z)[z \prec y \rightarrow \neg X(z)]]]$

WICHTIG: $\mathcal{N}_n$ ist ein infiniter, fix n-verzweigender Baum. Wir sind aber an definierbaren, variabel verzweigenden Teilbäumen, deren Knoten mit Elementen von **K** benannt und mit Elementen von **P** "gelabeled" werden können, interessiert. Daher gilt es für uns immer, daß, wenn ein Objekt eine gegebene Deskription ϕ erfüllt, dies relativ zu einer Menge von Knoten X ist, d.h.

$$\mathcal{N}_n \models (\exists X)[\phi(X)].$$

Im folgenden werden wir diese Menge unterschlagen, aber bitte bedenkt, daß alle vorkommenden Variablen Elemente oder Teilmengen dieser Menge sein müssen. Im Prinzip muß jede WFF aus $\mathcal{L}_{MSO}$ in die Sprache von n Nachfolgerfunktionen übersetzt werden, so daß die Bäume den erfüllenden Variablenzuweisungen entsprechen!

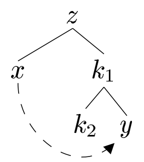
3 P&P-artige Anwendung in der Linguistik

3.1 Nichtlokale Bedingungen

(4)

$\mathsf{AC\text{-}Com}(x, y) \overset{def}{\Longleftrightarrow}$
% x c-commandiert y:
$(\forall z)[z \lhd^+ x \Rightarrow z \lhd^+ y] \wedge \neg(x \lhd^* y) \wedge$
% y c-commandiert x nicht:
$\neg((\forall z)[z \lhd^+ y \Rightarrow z \lhd^+ x] \wedge \neg(y \lhd^* x)) \wedge$
% x kommt vor y:
$x \prec y$

(5)



(6)

$\mathsf{LC}(x, y) \overset{def}{\Longleftrightarrow}$
$\mathsf{AC\text{-}Com}(x, y) \wedge$
% Es gibt kein k_1 mit Eigenschaft P:
$(\neg\exists k_1)[k_1 \in P \wedge$
% derart, daß es zwischen x und y interveniert:
$(\exists z)[z \lhd x \wedge z \lhd^+ k_1 \wedge k_1 \lhd^+ y]]$

3.2 Eine Mini-Grammatik

(7)
$$(\forall x)[\mathsf{Lex}(x) \vee \mathsf{Rule}(x)]$$

(8)
$$\mathsf{Lex}(x) \stackrel{def}{\Longleftrightarrow} \begin{array}{lccl} & (C(x) & \Rightarrow & weil) \\ \vee & (V(x) & \Rightarrow & schlug) \\ \vee & (N(x) & \Rightarrow & (Schuft \vee Hund)) \\ \vee & (D(x) & \Rightarrow & (der \vee den)) \end{array}$$

(9)
$$\mathsf{Rule}(x) \stackrel{def}{\Longleftrightarrow} \begin{array}{l} (\exists y,z)[(C(x) \Rightarrow x \lhd y \wedge x \lhd z \wedge C(y) \wedge V(z) \wedge y \prec z) \\ \quad \vee (V(x) \Rightarrow x \lhd y \wedge x \lhd z \wedge D(y) \wedge V(z) \wedge y \prec z) \\ \quad \vee (D(x) \Rightarrow x \lhd y \wedge x \lhd z \wedge D(y) \wedge N(z) \wedge y \prec z)] \end{array}$$

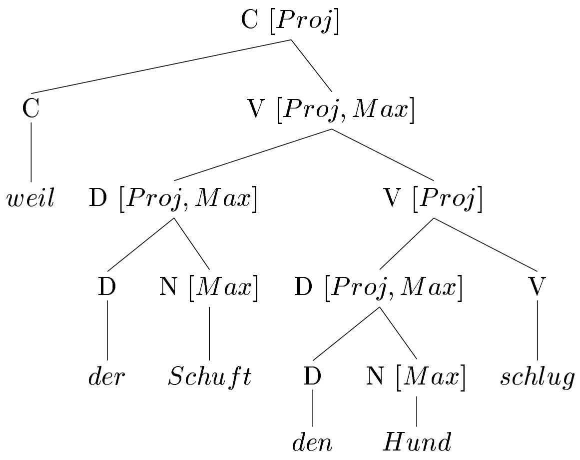
(10)
$$\mathsf{Cat}(x,y) \stackrel{def}{\Longleftrightarrow} \begin{array}{lccl} & (C(x) & \Leftrightarrow & C(y)) \\ \vee & (V(x) & \Leftrightarrow & V(y)) \\ \vee & (N(x) & \Leftrightarrow & N(y)) \\ \vee & (D(x) & \Leftrightarrow & D(y)) \end{array}$$

(11)
$$\begin{array}{l} (\forall x)[(\exists y,z)[x \lhd y \wedge x \lhd z \wedge \neg(y \approx z) \wedge (\forall w)[x \lhd w \Rightarrow (w \approx y \vee w \approx z)] \wedge \\ \qquad \neg\mathsf{Lex}(x)] \\ \quad \vee \neg(\exists y)[x \lhd y]\ ] \end{array}$$

(12)
$$(\forall x)[(\exists y)[x \lhd y \wedge \mathsf{Cat}(x,y)] \Leftrightarrow Proj(x)]$$

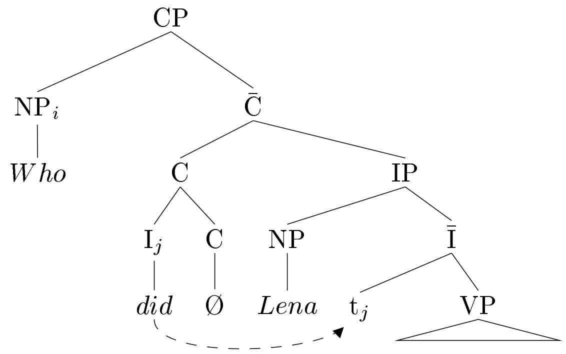
(13)
$$(\forall x)[(\exists y)[y \lhd x \wedge \neg\mathsf{Cat}(x,y)] \Leftrightarrow Max(x)]$$

(14)

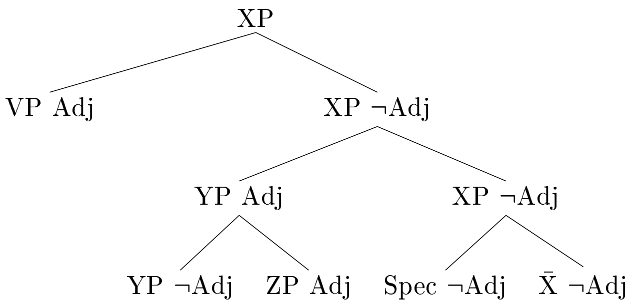


3.3 Komplexe Datenstrukturen

(15)



(16)



(17) $$\mathsf{Con}(X) \overset{def}{\Longleftrightarrow} (\forall x,y,z)[(X(x) \wedge X(y) \wedge x \lhd^* z \wedge z \lhd^* y) \Rightarrow X(z)]$$

(18) $$\mathsf{Path}(X) \overset{def}{\Longleftrightarrow} \mathsf{Con}(X) \wedge (\forall x,y)[(X(x) \wedge X(y)) \Rightarrow (x \lhd^* y \vee y \lhd^* x)]$$

(19) $$\mathsf{F.Eq}(x,y) \overset{def}{\Longleftrightarrow} (N(x) \Leftrightarrow N(y)) \wedge (V(x) \Leftrightarrow V(y)) \wedge \ldots$$

(20) $\mathsf{Comp}(X) \overset{def}{\Longleftrightarrow}$

$\mathsf{Path}(X)$
% alle Elemente haben die gleichen Merkmale:
$\wedge \quad (\forall x,y)[(X(x) \wedge X(y)) \Rightarrow \mathsf{F.Eq}(x,y)]$
% wenn es zwei Elemente gibt:
$\wedge \quad (\forall x,x')(\exists y)(\forall z)[(X(x) \wedge X(x') \wedge x \lhd x') \Rightarrow$
% dann gibt es ein Adjunkt:
$(\neg Adj(x') \wedge x \lhd y \wedge \neg(y \approx x') \wedge Adj(y)$
% und es gibt nur zwei Töchter:
$\wedge\, (x \lhd z \Rightarrow (z \approx x' \vee z \approx y)))]$

(21) $\mathsf{Cat}(X) \overset{def}{\Longleftrightarrow}$

$\mathsf{Comp}(X)$
% X ist die maximale Komponente:
$\wedge \quad (\forall Y)[(X \subseteq Y \wedge \neg(Y \subseteq X)) \Rightarrow \neg\mathsf{Comp}(Y)]$

(22) $$\mathsf{Cat}(X,x) \overset{def}{\Longleftrightarrow} \mathsf{Cat}(X) \wedge X(x)$$

(23) $$\mathsf{Cat}(x,y) \overset{def}{\Longleftrightarrow} (\exists X)[\mathsf{Cat}(X,x) \wedge \mathsf{Cat}(X,y)]$$

(24) $$(\forall x)[Adj(x) \Rightarrow (\exists y, z, Y)[y \lhd x \wedge y \lhd z \wedge \mathsf{Cat}(Y) \wedge Y(y) \wedge Y(z)]]$$

(25) $$\mathsf{Domi}(x,y) \stackrel{def}{\Longleftrightarrow} (\forall x')[\mathsf{Cat}(x,x') \Rightarrow x' \lhd^{+} y]$$

(26) $$\mathsf{Excl}(x,y) \stackrel{def}{\Longleftrightarrow} (\forall x')[\mathsf{Cat}(x,x') \Rightarrow \neg(x' \lhd^{*} y)]$$

(27) $$\mathsf{Incl}(x,y) \stackrel{def}{\Longleftrightarrow} \neg\mathsf{Excl}(x,y)$$

(28) $$\mathsf{Prec}(x,y) \stackrel{def}{\Longleftrightarrow} \mathsf{Excl}(x,y) \wedge \mathsf{Excl}(y,x) \wedge x \prec y$$

(29) $$\begin{aligned}\mathsf{Son}(x,y) \stackrel{def}{\Longleftrightarrow} & \\ & \mathsf{Domi}(x,y) \\ \wedge\ & \neg(\exists z)[\ (\mathsf{Excl}(z,x) \wedge \mathsf{Domi}(z,y)) \\ & \qquad \vee\ (Adj(z) \wedge \mathsf{Excl}(z,x) \wedge \mathsf{Incl}(z,y)]\end{aligned}$$

(30) $$\begin{aligned}\mathsf{C\text{-}Com}(x,y) \stackrel{def}{\Longleftrightarrow} & \\ & \neg\mathsf{Domi}(x,y) \wedge \neg\mathsf{Domi}(y,x) \\ \wedge\ & (\forall z)[\mathsf{Domi}(z,x) \Rightarrow \mathsf{Domi}(z,y)]\end{aligned}$$

(31) $$\begin{aligned}\mathsf{M\text{-}Com}(x,y) \stackrel{def}{\Longleftrightarrow} & \\ & \neg\mathsf{Domi}(x,y) \wedge \neg\mathsf{Domi}(y,x) \\ \wedge\ & (\forall z)[(Bar2(z) \wedge \mathsf{Domi}(z,x)) \Rightarrow \mathsf{Domi}(z,y)]\end{aligned}$$

(32) $$\mathsf{AC\text{-}Com}(x,y) \stackrel{def}{\Longleftrightarrow} \mathsf{C\text{-}Com}(x,y) \wedge \neg\mathsf{C\text{-}Com}(y,x) \wedge \mathsf{Prec}(x,y)$$

3.4 Weitere Anwendungen

1. Merkmalsbäume (bzw. Records): Niehren and Podelski (1992) und Ayari et al. (1997)

2. Hardware Verification: Basin and Klarlund (1995)

3. TAG Deskriptionslogiken: Rogers and Vijay-Shanker (1992), Rogers and Vijay-Shanker (1995) und Rogers (1994)

4. ...

4 Der Entscheidbarkeitsbeweis

Theorem 4.1 (Rabin) *SnS für $n \leq \omega$ ist entscheidbar.*

In diesem Abschnitt schränken wir uns wie folgt ein:

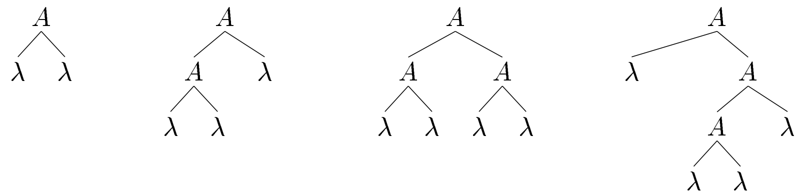
- nur binäre Bäume
- nur endliche Mengen
- keine Konstanten
- keine Individuenvariablen

Daher bezeichnen wir die Sprache $\mathcal{L}_{MSO}$ im Folgenden mit WS2S.

4.1 Baumautomaten

Definition 4.2 *Sei $\Sigma = \Sigma_0 \cup \Sigma_2$ ein Alphabet mit $\Sigma_0 = \{\lambda\}$ und Σ_2 einer Menge von binären Operationssymbolen.* ***Binäre Bäume über*** *Σ sind dann genau die Terme T_Σ über diesem Alphabet. Genauer, λ ist der leere Baum und $\sigma(t_1, t_2)$, für $\sigma \in \Sigma_2$ und $t_1, t_2 \in T_\Sigma$, ist der Baum mit Label σ und Teilbäumen t_1, t_2.*

(33)



Definition 4.3 *Ein* ***deterministischer (bottom-up) Baumautomat*** *$\mathfrak{A}$ auf binären Bäumen ist ein Tupel $\langle A, \Sigma, a_0, F, \alpha\rangle$ mit A der Zustandsmenge, $a_0 \in A$ dem Startzustand, $F \subseteq A$ den Finalzuständen und $\alpha : (A \times A \times \Sigma) \to A$ der Übergangsfunktion. Die Übergangsfunktion induziert einen induktiv definierten Homomorphismus:*

- $h_\alpha(\lambda) = a_0$
- $h_\alpha(\sigma(t_1, t_2)) = \alpha(h_\alpha(t_1), h_\alpha(t_2), \sigma)$.

Ein Automat $\mathfrak{A}$ ***akzeptiert*** *einen Baum t genau dann, wenn $h_\alpha(t) \in F$. Wir definieren die* ***Sprache eines Automaten*** *$\mathfrak{A}$ mit $T(\mathfrak{A}) = \{t \mid h_\alpha(t) \in F\}$.*

(34)

$$\mathfrak{A} = \langle \{a_0, a_1\}, \{A, B\}, a_0, \{a_0\}, \alpha\rangle$$

$$\begin{array}{ll}
\alpha(a_0, a_0, A) = a_0 & \alpha(a_0, a_1, A) = a_1 \\
\alpha(a_1, a_0, A) = a_1 & \alpha(a_1, a_1, A) = a_1 \\
\alpha(a_0, a_0, B) = a_1 & \alpha(a_0, a_1, B) = a_1 \\
\alpha(a_1, a_0, B) = a_1 & \alpha(a_1, a_1, B) = a_1
\end{array}$$

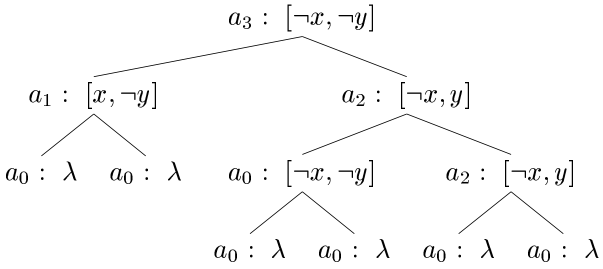
(35)

$$\begin{gathered}
\mathfrak{A} = \langle A, \Sigma, a_0, F, \alpha\rangle \\
A = \{a_0, a_1, a_2, a_3, a_4\} \\
\Sigma = \{[x, y], [x, \neg y], [\neg x, y], [\neg x, \neg y]\} \\
F = \{a_3\}
\end{gathered}$$

$$\begin{array}{ll}
\alpha(a_0, a_0, [\neg x, \neg y]) = a_0 & \alpha(a_0, a_0, [x, \neg y]) = a_1 \\
\alpha(a_0, a_0, [\neg x, y]) = a_2 & \alpha(a_0, a_2, [\neg x, \neg y]) = a_2 \\
\alpha(a_0, a_3, [\neg x, \neg y]) = a_3 & \alpha(a_2, a_0, [\neg x, \neg y]) = a_2 \\
\alpha(a_1, a_2, [\neg x, \neg y]) = a_3 & \alpha(a_3, a_0, [\neg x, \neg y]) = a_3
\end{array}$$

alle anderen Übergänge sind zu a_4

(36)



Lemma 4.4 ***"Emptiness"*** *einer von einem Automaten erkannten Sprache ist entscheidbar durch die folgende Fixpunktkonstruktion:*

1. $R := \{a_0\}$, $R' := \emptyset$.
2. *For all* $(a_i, a_j) \in R \times R$, *for all* $\sigma \in \Sigma$,
   $R' := R' \cup \{\alpha(a_i, a_j, \sigma)\}$.
3. *If* $R' - R = \emptyset$ *then return* R,
   else $R := R \cup R'$, *go to step 2.*

R enthält die erreichbaren Zustände eines Automaten. R' enthält die in einer Schleife neu gefundenen Zustände. Die Sprache ist leer, wenn wir einen Finalzustand in R' finden. Der Algorithmus terminiert nach höchstens i Schritten (i die Anzahl der Zustände).

Definition 4.5 *Mengen von Bäumen, die von Baumautomaten erkannt werden, nennt man* ***erkennbar****.*

Lemma 4.6 *Erkennbare Mengen sind abgeschlossen unter Konjunktion, Disjunktion, Negation (Komplement), Projektion und Zylindrifizierung, da es für sie entsprechende bezeugende Automatenkonstruktionen gibt. Weitere wohldefinierte Konstruktionen sind Determinisierung und Minimierung.*

Theorem 4.7 *WFF aus WS2S können durch Baumautomaten repräsentiert werden, die die relevanten Variablenzuweisungen erkennen:*

$$\mathcal{N}_2 \models \phi[t] \Leftrightarrow t \in T(\mathfrak{A}^{\phi})$$

4.2 Der Beweis für $\mathcal{L}^2$

Theorem 4.8 *Wenn ϕ und ψ erkennbare Relationen definieren, dann auch:*

$$\phi \wedge \psi \qquad\qquad \neg\phi \qquad\qquad (\exists X)\phi.$$

Der Beweis wird geführt, in dem man die entsprechenden Automatenkonstruktionen für $\mathfrak{A}^{(\phi\wedge\psi)}, \mathfrak{A}^{(\neg\phi)}$ und $\mathfrak{A}^{((\exists X)\phi)}$ für gegebene $\mathfrak{A}^{(\phi)}$ und $\mathfrak{A}^{(\psi)}$ präsentiert.
Als Beispiel hier die Konjunktion: $U = T(\mathfrak{A})$ und $V = T(\mathfrak{B})$ seien erkennbare Mengen von Bäumen.

$$\begin{aligned} \mathfrak{A} \times \mathfrak{B} &= (A \times B, \Sigma, (a_0, b_0), (A_F \times B_F), \gamma) \\ \Sigma &= \Sigma_{\mathfrak{A}} \cup \Sigma_{\mathfrak{B}} \\ \gamma((a_1, b_1), (a_2, b_2), \sigma) &= (\alpha(a_1, a_2, \sigma), \beta(b_1, b_2, \sigma)) \end{aligned} \tag{37}$$

Dann gilt

$$U \cap V = T(\mathfrak{A} \times \mathfrak{B}) \tag{38}$$

Theorem 4.9 *Die Relationen, die durch die folgenden Formeln definiert werden, sind erkennbar:*

$$x \lhd^* y \qquad x \lhd y \qquad x \prec y \qquad x \approx y \qquad x \in X.$$

Der Beweis wird geführt, in dem man Automaten für die Relationen in Theorem 4.9 angibt, z.B. für reflexive Dominanz in (39).

$$\begin{array}{ll} \multicolumn{2}{c}{\mathfrak{A} = \langle \{a_0, a_1, a_2, a_3\}, \Sigma, a_0, \{a_2\}, \alpha \rangle} \\ \\ \alpha(a_0, a_0, [\neg x, \neg y]) = a_0 & \alpha(a_0, a_1, [\neg x, \neg y]) = a_1 \\ \alpha(a_1, a_0, [\neg x, \neg y]) = a_1 & \alpha(a_0, a_2, [\neg x, \neg y]) = a_2 \\ \alpha(a_2, a_0, [\neg x, \neg y]) = a_2 & \alpha(a_0, a_0, [\neg x, y]) \ = a_1 \\ \alpha(a_1, a_0, [x, \neg y]) \ = a_2 & \alpha(a_0, a_1, [\neg x, \neg y]) = a_2 \\ \multicolumn{2}{c}{\text{alle anderen Übergänge sind zu } a_3} \end{array} \tag{39}$$

4.3 Generative Kapazität

Theorem 4.10 *Die Klasse von String-Sprachen, die die Yields von erkennbaren Mengen (Bäumen) formen, entspricht genau der der kontextfreien Sprachen.*

Lemma 4.11 *Prinzipien bzw. Constraints, die man in WS2S definieren kann, lizensieren genau die schwach kontextfreien Sprachen.*

Die erkennbaren Mengen formen eine echte Obermenge der lokalen Mengen (der Mengen, die durch Derivationen von kontextfreien Grammatiken erzeugt werden). Ein Beispiel für eine erkennbare Menge, die nicht lokal ist, ist die Menge aller binärer Bäume in denen genau ein Knoten mit B, alle anderen mit A benannt sind. Diese Menge ist nicht lokal, da eine CFG, wenn sie ein Symbol einmal auf eine bestimmte Art expandieren kann, dieses Symbol immer auf diese Art expandieren kann. Es gibt einen Ausweg, der darin besteht, daß man die Kategorien der CFG verfeinert (à la GPSG mit Slashes). Eine Menge von Bäumen ist eine Projektion einer lokalen Menge, wenn man ihre Label so in endlich viele Sub-Label verfeinern kann, daß man Knoten die unterschiedlich expandieren unterscheiden kann.

Lemma 4.12 *Die erkennbaren Mengen sind Projektionen der lokalen Mengen.*

4.4 Anwendungen und Probleme

Nicht-Definierbarkeit: Da wir definierbare Prädikate zur Signatur hinzunehmen können, ohne die Entscheidbarkeit zu verlieren, ist eine Art Nicht-Definierbarkeit zu zeigen, die Nicht-Entscheidbarkeit der expandierten Signatur zu zeigen. Das wird dann (letztendlich) durch Reduzierung auf das Halteproblem geschehen. Beispiele für nicht-definierbare Relationen:

- freie Indizierung: wird in GB oft angenommen, ist aber umstritten. Der Formalismus "löst" diese Streitfrage.
- Teilbaum-Isomorphie: Wir können keine Merkmalsstrukturen erfassen, da deren zentrales Mittel gerade diese Isomorphie als Haupterklärungsmittel einsetzen
- Gleiche Tiefe von Knoten im Baum
- $s : s(w) \longmapsto 0w$
- gleiches Labeling von zwei Yields

Implementierung: Der Beweis ist effektiv, d.h. wir können ihn so implementieren, wie er hier präsentiert ist. Es resultiert ein Compiler/Theorembeweiser. Das Programm compiliert eine Formel in einen Automaten. Wenn dieser nicht leer ist, dann stellt der existentielle Abschluß der Relation ein Theorem dar. Erzeugte Automaten repräsentieren die möglicherweise infinite Zahl von erfüllenden Variablenzuweisungen. Wenn man die Automaten als Generator verwendet und die Struktur des Generierungsprozesses verwendet, ergeben sich die erkannten Bäume.

Parsing als Theorembeweisen: Wie standardmäßig in erfüllbarkeitsbasierten Ansätzen besteht Parsing einfach aus der Konjunktion von einer Grammatikformel/einem Grammatikautomaten mit einer Inputformel/einem Inputautomaten Johnson (1994, 1995). Wenn wir so einen Automaten haben und die erkannte Sprache nicht leer ist, haben wir den Input erkannt. Wir können ihn "laufen lassen" und damit die Parse-Bäume generieren. Naïve kann man dann sagen, gegeben ein

Automat $Gram(X)$ und ein Input *John sees Mary*, die zu lösende Formel sieht wie folgt aus:

(40) $(\exists x, y, z \in X)[x \in John \wedge y \in Sees \wedge z \in Mary \wedge x \prec y \prec z \wedge Gram(X)]$

Aber es gibt die folgenden Probleme:

1. Leider ist die Konstruktion/Compilierung von nicht-elementarer Komplexität. Problematisch sind hierbei
   (a) Determinisierung: sie ist nötig für jeden "Quantorenwechsel", da Projektion nichtdeterministische Automaten erzeugt, Negation aber deterministischen Input braucht; und
   (b) Anzahl der freien Variablen: sie müssen in das Automatenalphabet kodiert werden, was zu einer kombinatorischen Explosion führt, wenn alle Permutationen ihrer Vorkommen berücksichtigt werden müssen (siehe Beispiel 35).
2. Natürliche Sprachen sind nicht generell kontextfrei (??) (Shieber 1985, Huybregts 1984), daher stellt die Kontextfreiheit ein weiteres Problem dar.

Ansätze/Vorschläge diese Probleme zu lösen:

- Mönnich (1997) schlägt eine algebraisch basierte Kodierung vor, in der Funktionen als Knotenlabel behandelt werden, so daß die akzeptierten Strukturen später durch einen Ausführungshomomorphismus erst in die gewünschten Bäume transformiert werden müssen;
- Morawietz (1997) bettet WS2S in ein "constraint logic programming" Schema ein, wodurch induktive Definitionen zum Bestandteil der Sprache werden. Dadurch kann man Programme schreiben, die Relationen derart definieren, daß sowohl der Compilierungsprozeß gesteuert, als auch die Kontextfreiheitsbarriere überwunden werden kann.
- Morawietz and Cornell (1997a): Redefinition des Problems; Anstatt einer *vollständigen* Grammtik wird eine *partielle* Grammatik betrachtet, die je nach Input ergänzt wird (ein Beispiel sind Indizes).
- ...

5 Zusammenfassung

1. Stärken von MSO Logik:
   (a) Vorteile aller Logik- bzw. Constraint-basierter Systeme
   (b) erlaubt die Überbrückung von einem linguistisch geeigneten Formalismus (Logik) zur computerlinguistischen Anwendung (Automaten)
   (c) Entscheidbar
   (d) Ausdruckskraft (z.B. Endlichkeit ist definierbar): stärker als Logik erster Ordnung, da aber 1c, schwächer als Logik erster Ordnung
   (e) volle Negation/Quantifikation: nicht lokale Beschränkungen
   (f) flexibel: Signatur kann angepaßt werden (Frank and Vijay-Shanker 1995, z.B. f"ur Theorien mit c-commando als Basisrelation wir in)
   (g) keine unnötige Struktur auf Merkmalen

(h) Deskriptive Komplexität

2. Schwächen:

    (a) Deskriptive Komplexität

    (b) Komplexität der Automatengenerierung

Literatur

Ayari, A., Basin, D. and Podelski, A. (1997). LISA: A specification language based on ws2s, Ms, Universität Freiburg. Submitted to CSL'97.

Basin, D. and Klarlund, N. (1995). Hardware verification using monadic second-order logic, *Computer-Aided Verification (CAV '95)*, Vol. 939 of *LNCS*, LNCS 939, Springer, pp. 31–41.

Blackburn, P. and Meyer-Viol, W. (1994). Linguistics, logic, and finite trees, *Report CS-R9412*, CWI, Amsterdam.

Blackburn, P., Gardent, C. and Meyer-Viol, W. (1993). Talking about trees, *Proceedings of the 6th EACL*, pp. 21–29.

Blackburn, P., Meyer-Viol, W. and de Rijke, M. (1995). A proof system for finite trees, *Technical Report 67*, Computational Linguistics at the University of the Saarland (CLAUS).

Brody, M. (1995). *Lexico-Logical Form*, MIT Press, Cambridge, MA.

Büchi, J. R. (1960). Weak second-order arithmetic and finite automata, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **6**: 66–92.

Cornell, T. (1996). Model-theoretic syntax, *Glot International*.

Doner, J. (1970). Tree acceptors and some of their applications, *Journal of Computer and System Sciences* **4**: 406–451.

Elgot, C. C. (1961). Decision problems of finite automata design and related arithmetics, *Trans. Amer. Math. Soc.* **98**: 21–51.

Frank, R. and Vijay-Shanker, K. (1995). C-command and grammatical primitives, Presentation at the 18th GLOW Colloquium. University of Tromsø.

Gécseg, F. and Steinby, M. (1984). *Tree Automata*, Akadémiai Kiadó, Budapest.

Henriksen, J. G., Jensen, J., Jørgensen, M., Klarlund, N., Paige, R., Rauhe, T. and Sandhol, A. (1995). MONA: Monadic second-order logic in practice, *in* Brinksma, Cleaveland, Larsen, Margaria and Steffen (eds), *Tools and Algorithms for The Construction and Analysis of Systems: International Workshop*, TACAS '95, Selected Papers, Lecture Notes in Computer Science 1019, Springer, pp. 89–110.

Huybregts, R. (1984). The weak inadequacy of context-free phrase structure grammars, *in* de Haan, Trommelen and Zonneveld (eds), *Van Periferie naar Kern*, Foris, Dordrecht, pp. 81–99.

Johnson, M. (1994). Two ways of formalizing grammars, *Linguistics and Philosophy* **17**: 221–248.

Johnson, M. (1995). Constraint-based natural language parsing, ESSLLI '95, Barcelona, Course notes.

Kracht, M. (1992). The theory of syntactic domains, *Technical Report 75*, Department of Philosophy, University of Utrecht.

Kracht, M. (1993). Mathematical aspects of command relations, *Proceedings of the 6th EACL*, pp. 240–249.

Kracht, M. (1995). Syntactic codes and grammar refinement, *Journal of Logic, Language and Information* **4**: 41–60.

Mönnich, U. (1997). On cloning context-freeness, *Ms.*, SFB 340, Universität Tübingen. To appear in H.-P. Kolb & U. Mönnich *The Mathematics of Syntactic Structure*.

Morawietz, F. (1997). Monadic second order logic, tree automata and constraint logic programming, *Arbeitspapiere des SFB 340 86*, SFB 340, Universität Tübingen. To appear in H.-P. Kolb & U. Mönnich *The Mathematics of Syntactic Structure*.

Morawietz, F. and Cornell, T. (1997a). Approximating principles and parameters grammars with mso tree logics. Four page abstract. Submitted to LACL '97.

Morawietz, F. and Cornell, T. L. (1997b). On the recognizability of relations over a tree definable in a monadic second order tree description language, *Arbeitspapiere des SFB 340 85*, SFB 340, Universität Tübingen.

Morawietz, F. and Cornell, T. L. (1997c). Representing constraints with automata, *To appear in: Proceedings of the ACL/EACL Conference*, Association for Computational Linguistics, Madrid, Spain.

Niehren, J. and Podelski, A. (1992). Feature automata and recognizable sets of feature trees, *in* M.-C. Gaudel and J.-P. Jouannaud (eds), *Proccedings of the 4th International Joint Conference on Theory and Practice of Software Development*, Springer, LNCS 668, Orsay, France, pp. 356–375.

Rabin, M. O. (1969). Decidability of second-order theories and automata on infinite trees, *Transactions of the American Mathematical Society* **141**: 1–35.

Rabin, M. O. (1970). Weakly definable relations and special automata, *in* Y. Bar-Hillel (ed.), *Mathematical Logic and Foundations of Set Theory*, North-Holland, Amsterdam, pp. 1–23.

Rizzi, L. (1990). *Relativized Minimality*, MIT Press.

Rogers, J. (1994). *Studies in the Logic of Trees with Applications to Grammar Formalisms*, PhD thesis, University of Delaware, Dep. of Computer & Information Sciences, Newark, DE 19716. Published as Technical Report No. 95-04.

Rogers, J. (1996a). A model-theoretic framework for theories of syntax, *Proc. of the 34th Annual Meeting of the ACL*, Santa Cruz, USA.

Rogers, J. (1996b). Strict lt_2: Regular :: Local : Recognizable, Ms, To appear in Proceedings of LACL '96.

Rogers, J. (1997). On descriptive complexity, language complexity, and GB, *in* P. Blackburn and M. de Rijke (eds), *Specifying Syntactic Structures*, CSLI Publications, Stanford, Ca., p. .

Rogers, J. (To appear). *A Descriptive Approach to Language-Theoretic Complexity*, Studies in Logic, Language, and Information, CSLI Publications. Available via the www at http://www.cis.upenn.edu/˜jrogers/home.html.

Rogers, J. and Vijay-Shanker, K. (1992). Reasoning with descriptions of trees, *30th Annual Meeting of the ACL: Proceedings of the Conference*, Association for Computational Linguistics, pp. 72–80.

Rogers, J. and Vijay-Shanker, K. (1995). Obtaining trees from their descriptions: An application to tree-adjoining grammars, *Computational Intelligence* **10**: 401–421.

Rounds, W. C. (1970a). Mappings and grammars on trees, *Mathematical Systems Theory* **4**: 257–287.

Rounds, W. C. (1970b). Tree-oriented proofs of some theorems on context-free and indexed languages, *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing*, pp. 109–116.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language, *Linguistics and Philosophy* **8**: 333–343.

Stabler, E. P. (1992). *The Logical Approach to Syntax: Foundations, Specifications and Implementations*, MIT Press, Cambrigde, Massachusetts.

Thatcher, J. (1967). Characterizing derivation trees of contextfree grammars through a generalization of finite automata theory, *J. Comput. System Sci.* **1**: 317–322.

Thatcher, J. W. and Wright, J. B. (1968). Generalized finite automata theory with an application to a decision problem of second-order logic, *Mathematical Systems Theory* **2**(1): 57–81.