Konstruktion eines inkrementellen Auswerters für geordnete Makroattributgrammatiken

Diplomarbeit

Bernd Jokubeit

Technische Universität Dresden
Fakultät Informatik
Lehrstuhl Grundlagen der Programmierung

Dresden, den 6. Dezember 1996

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Makroattributgrammatiken sind ein Modell zur Beschreibung der Semantik der Sätze kontextfreier Sprachen, das aus einer Verallgemeinerung der Attributgrammatiken resultiert. Analog zu den Attributgrammatiken wird die Semantik jedes Satzes durch die Attributierung seines Ableitungsbaums bestimmt. Analog zu den Attributgrammatiken wird die Attributierung eines Ableitungsbaums durch die Zuordnung von Attributen zu jedem Nichtterminalsymbol einer kontextfreien Grammatik und durch die Zuordnung von semantischen Regeln zu jeder Produktion der Grammatik spezifiziert. Dabei definieren die semantischen Regeln einer Produktion die funktionalen Abhängigkeiten zwischen den Attributen der Nichtterminalsymbole, die in der Produktion vorkommen. Jedoch gestatten die Makroattributgrammatiken im Gegensatz zu den Attributgrammatiken die Spezifikation von *Makroattributen*, deren Instanzen *Funktionen* als Werte zugeordnet werden können. Damit ist der Typ des Wertes einer Instanz eines Makroattributs nicht mehr *ein atomarer Wert aus dem semantischen Bereich*, sondern vielmehr *eine Funktion über Mengen aus dem semantischen Bereich*. Dementsprechend haben die semantischen Regeln eine verallgemeinerte Struktur, die die Deklaration von Funktionen erlaubt, die später, in einem Ableitungsbaum, den Instanzen der Attribute zugeordnet werden. Die rechten Seiten der semantischen Regeln jeder Produktion sind parametrisierte Terme über einer Symbolmenge, die sich aus Funktionssymbolen des semantischen Bereichs *und* aus „Attributvorkommenssymbolen" der Produktion zusammensetzen.

Dieser Schritt der Verallgemeinerung der Attributierung nebst grundlegender Eigenschaften, die das verallgemeinerte Modell besitzt, werden in der Arbeit von Kühnemann und Vogler [KV94] sowie in der Arbeit von Adamski und Vogler [AV96] untersucht. Aus der Arbeit von Adamski und Vogler geht unter anderem hervor, daß die semantischen Regeln der Produktion von Makroattributgrammatiken, trotz ihrer reicheren Struktur, die gleiche Abhängigkeitsrelation auf den Attributvorkommen induzieren wie die semantischen Regeln aus einer Attributgrammatik: Das Attributvorkommen auf der linken Seite einer semantischen Regel r ist von allen Attributvorkommen abhängig, die im Term auf der rechten Seite von r auftreten. Aus r lassen sich jedoch keine Abhängigkeiten zwischen den Attributvorkommen auf der rechten Seite von r ableiten.

Abbildung 1.1 illustriert diesen Gedanken, indem sie die Abhängigkeiten zwischen den Attributvorkommen $\langle u, 0\rangle$, $\langle u, 1\rangle$ und $\langle u, 2\rangle$ einer Produktion $p : L \to LL$ zeigt, die aus einer semantischen Regel r von p resultieren, die folgende Form hat:

$$r : \langle u, 0\rangle(y) = \langle u, 1\rangle(\langle u, 2\rangle(y)).$$

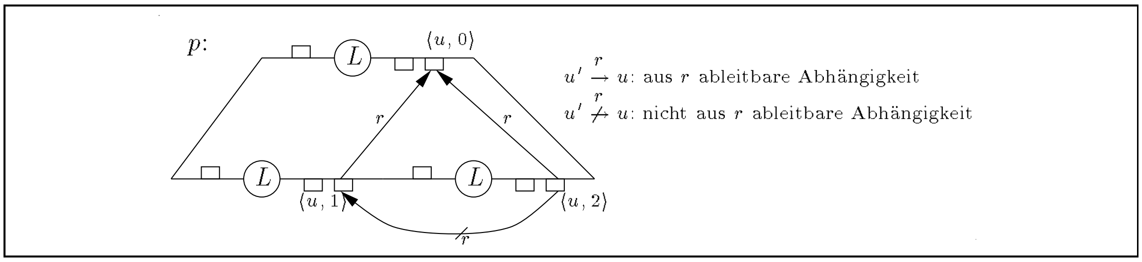


Abbildung 1.1: Abhängigkeiten zwischen Makroattributvorkommen

Diese Eigenschaft der semantischen Regeln ist von entscheidender Bedeutung für die Konstruktion von Auswertern für Makroattributgrammatiken, eröffnet sie doch die Möglichkeit, die existierenden Auswerter für Attributgrammatiken, die auf Ordnungseigenschaften der induzierten Abhängigkeitsrelationen beruhen, dahingehend zu untersuchen, ob sie sich für die Auswertung von Makroattributgrammatiken anpassen lassen.

Mit dieser Überlegung beginnt die Untersuchung inkrementeller Auswerter für geordnete Makroattributgrammatiken.

Intuitiv heißt eine Makroattributgrammatik *MAG* geordnet, wenn für jedes Nichtterminalsymbol N von *MAG* eine Ordnung über den Attributen von N aus den Abhängigkeitsrelationen der Attributvorkommen konstruierbar ist. Damit ist die Klasse der geordneten Makroattributgrammatiken eine Verallgemeinerung der Klasse der geordneten Attributgrammatiken, die in der Arbeit von Kastens [Kas80] eingeführt wurden.

Da die Zugehörigkeit einer Makroattributgrammatik zur Klasse der geordneten Attributgrammatiken auf der Struktur der Abhängigkeitsrelationen basiert, wird sowohl der iterative Auswerter, den Kastens für geordnete Attributgrammatiken in [Kas80] angibt, als auch der inkrementelle Auswerter, den Reps und Teitelbaum in [RT89] aus Kastens Algorithmus entwickeln, bezüglich der Verwendbarkeit zur Auswertung geordneter Makroattributgrammatiken untersucht.

Beide Algorithmen berechnen die Werte der Attributinstanzen eines Ableitungsbaums, indem sie den Ableitungsbaum traversieren und jeweils diejenigen Attributinstanzen berechnen, die höchstens von bereits früher berechneten Instanzen abhängig sind. Die Schwierigkeit bei der Adaption der beiden Algorithmen ist weniger darin zu suchen, die Traversierung zu übertragen, als vielmehr darin, *wie* die Instanzen der funktionswertigen Makroattribute zu berechnen und zu *repräsentieren* sind.

Die Suche nach geeigneten Repräsentationen für die Werte der Makroattributinstanzen resultiert aus der Notwendigkeit, die bereits früher berechneten Werte zu speichern, damit sie für die Berechnung der Werte weiterer Instanzen zur Verfügung stehen. Ein erster Ansatz, der in der vorliegenden Arbeit untersucht wird, führt zur Repräsentation der Werte von Makroattributinstanzen in Form von Termen, die sich aus den rechten Seiten der Instanziierungen der semantischen Regeln gewinnen lassen. Die Termrepräsentation ermöglicht die Adaption aller existierenden Attributauswerter, die die Ableitungsbäume traversieren. Erst bei der Adaption des inkrementellen Auswerters zeigt sich ein Problem der Termrepräsentation der Werte der Instanzen der Makroattribute: Bei der wiederholten Neuberechnung größerer Terme ist dieser Ansatz sehr schwerfällig.

Mit der Beschreibung des Konzeptes der *Termsegmentdekoration* und ihrer graphischen Repräsentation endet in der vorliegenden Arbeit die Untersuchung inkrementeller Auswerter für geordnete Makroattributgrammatiken.

Der Rest der Arbeit ist folgendermaßen gegliedert:

In Kapitel 2 sind allgemeine Grundlagen zusammengestellt.

In Kapitel 3 erfolgt die Einführung in die Konzepte zu den Makroattributgrammatiken. Dazu gehören neben der Definition der Makroattributgrammatiken hauptsächlich die Beschreibung der Konzepte, mit denen die Semantik der Sätze realisiert wird. Dazu werden die Termrepräsentation und die Attributauswerter definiert.

In Kapitel 4 werden die geordneten Makroattributgrammatiken definiert und der erste iterative Auswerter zur Berechnung der Werte der Instanzen von Makroattributen konstruiert.

In Kapitel 5 wird aus dem iterativen Auswerter der erste inkrementelle Auswerter für geordnete Makroattributgrammatiken konstruiert.

In Kapitel 6 wird der inkrementelle Auswerter aus Kapitel 5, der eine sehr direkte Adaption des Algorithmus von Reps und Teitelbaum darstellt, dahingehend erweitert, daß der resultierende Auswerter ebenfalls die Termstruktur der Werte der Instanzen berücksichtigt.

Kapitel 2

Grundlagen

Im Folgenden sind die grundlegenden Begriffe und Notationen zusammengefaßt, die zur Darstellung der in der vorliegenden Arbeit untersuchten Konzepte unerläßlich sind. Die benutzte Terminologie stützt sich auf die, die von Kühnemann und Vogler in [KV94] und von Adamski und Vogler in [AV96] verwendet und entwickelt wurde.

2.1 Allgemeine Notationen

Die leere Menge wird mit $\emptyset$ bezeichnet. Für eine beliebige Menge M bezeichnet $\mathcal{P}(M)$ die Menge aller Teilmengen von M und $|M|$ die Kardinalität von M. Für zwei beliebige Mengen M und N bezeichnen $M \cup N$ die Vereinigungsmenge, $M \cap N$ den Durchschnitt und $M \setminus N$ die Differenzmenge von M und N. Die Menge der natürlichen Zahlen wird mit $\mathbb{N}$, die Menge $\mathbb{N} \setminus \{0\}$ der positiven ganzen Zahlen mit $\mathbb{N}_+$ bezeichnet. Für zwei beliebige natürliche Zahlen $u, o \in \mathbb{N}$ bezeichnet $\{u, \ldots, o\}$ die Menge aller Zahlen $n \in \mathbb{N}$ für die $u \leq n \leq o$ gilt. Für ein beliebiges $k \in \mathbb{N}$ und für beliebige Mengen $M_1, \ldots, M_k$ bezeichnet $M_1 \times \ldots \times M_k = \{(m_1, \ldots, m_k) \mid m_i \in M_i\}$ das kartesische Produkt von $M_1, \ldots, M_k$; jede Teilmenge $R \subseteq M_1 \times \ldots \times M_k$ heißt k-stellige Relation. Für drei beliebige Mengen M_1, M_2 und M_3 und zwei beliebige binäre Relationen $R_1 \subseteq M_1 \times M_2$ und $R_2 \subseteq M_2 \times M_3$ bezeichnet $R_1 \circ R_2 \subseteq M_1 \times M_3$ die Komposition von R_1 und R_2. Für eine beliebige Menge M und eine beliebige Relation $R \subseteq M \times M$ bezeichnet R^+ die transitive Hülle von R und $R^* = R^+ \cup \{(a,a) \mid a \in M\}$ die reflexive transitive Hülle von R; für eine beliebige Teilmenge $N \subseteq M$ bezeichnet $R\lceil N\rceil = R \cap (N \times N)$ die Einschränkung von R auf die Teilmenge N von M. Das leere Wort wird mit ε bezeichnet. Ist Σ eine endliche Menge von Symbolen, so bezeichnet Σ^* die Menge aller endlichen Zeichenketten über Σ. Ist $w \in \Sigma^*$ ein Wort über Σ, so bezeichnet $|w|$ die Länge von w. Sind $L_1, L_2 \subseteq \Sigma^*$, so bezeichnet $L_1 \circ L_2 = \{w_1 \circ w_2 \mid w_1 \in L_1, w_2 \in L_2\}$ sowie $L_1 L_2 = L_1 \circ L_2$ und $w_1 w_2 = w_1 \circ w_2$ die Konkatenation[1]. Für eine beliebige Menge M bezeichnet $e_1; \ldots; e_l$ mit $l \in \mathbb{N}_+$ und $e_1, \ldots, e_l \in M$ eine Sequenz von Elementen aus M. Ist S eine Sequenz von Elementen aus M, so bezeichnet $|S|$ die Länge von S; gilt darüberhinaus für alle $i, j \in \{1, \ldots, |S|\}$, aus $i \neq j$ folgt $e_i \neq e_j$, dann bezeichnet $pos_S(e_i) = i$ die Position des Elements e_i in S.

[1] nach [CES88, Seite 302f.]

2.2 Gerichtete Graphen

Definition 2.1 (Gerichteter Graph)
Ein *gerichteter Graph* ist ein Tupel (V, E) bestehend aus

- einer endlichen nichtleeren Menge V, die als *Knotenmenge* bezeichnet wird, und
- einer Relation $E \subseteq V \times V$, die als *Kantenmenge* bezeichnet wird.

Für ein Paar $(v_1, v_2) \in E$ heißen $v_1 \in V$ ein *direkter Vorgängerknoten von* v_2 und $v_2 \in V$ ein *direkter Nachfolgerknoten von* v_1. Für ein Paar $(v_1, v_2) \in E^+ \setminus E$ heißen $v_1 \in V$ ein *indirekter Vorgängerknoten von* v_2 und $v_2 \in V$ ein *indirekter Nachfolgerknoten von* v_1. Geht die Richtung einer Kante $(v_1, v_2) \in E$ aus dem Kontext hervor, oder wird allgemein von der Richtung abstrahiert, werden v_1 und v_2 auch *Nachbarn* genannt.

Wird neben den Mengen V und E eine weitere Menge Σ von Symbolen und eine Abbildung $label : V \to \Sigma$ angegeben, die jedem Knoten ein Symbol $\sigma \in \Sigma$ zuordnet, so bezeichnet das Tupel $(V, E, \Sigma, label)$ einen *gerichteten beschrifteten Graphen.*

Seien $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ zwei gerichtete Graphen. Die Vereinigung von G_1 und G_2, bezeichnet mit $G_1 \cup G_2$, ist der Graph $(V_1 \cup V_2, E_1 \cup E_2)$.

$\square$

Definition 2.2 (Gerichteter azyklischer Graph)
Ein gerichteter Graph $G = (V, E)$ heißt genau dann *azyklisch*, wenn die transitive Hülle E^+ der Kantenrelation des Graphen kein Paar (v, v) mit $v \in V$ enthält.

$\square$

Definition 2.3 (Topologische Sortierung)
Sei $G = (V, E)$ ein gerichteter azyklischer Graph. Eine topologische Sortierung von G, bezeichnet mit $topsort(G)$, ist eine Sequenz aller Knoten des Graphen

$$topsort(G) = v_1; \ldots ; v_m \qquad m = |\, V \,|$$

für die gilt:

- $\{v_i \mid i \in \{1, \ldots, m\}$ und $v_i \in topsort(G)\} = V$
- Für alle $i, j \in \{1, \ldots, m\}$ und für alle $v_i, v_j \in V$ gilt: Ist $(v_i, v_j) \in E$, dann ist $i < j$.

$\square$

2.3 Sortierte Mengen und sortierte Bäume

Definition 2.4 (S- und $(S^* \times S)$-sortierte Menge)
Sei S eine endliche, nichtleere Menge (von Sortennamen). Eine *S-sortierte Menge* ist ein Tupel $(\Sigma, sort)$, wobei Σ eine Menge von Symbolen und $sort : \Sigma \to S$ eine Abbildung ist, die jedem Symbol $\sigma \in \Sigma$ eine Sorte $s \in S$ zuordnet. Eine *$(S^* \times S)$-sortierte Menge*

ist ein Tupel $(\Sigma, sort)$, wobei Σ eine Menge von Symbolen und $sort : \Sigma \to (S^* \times S)$ eine Abbildung ist, die jedem Symbol $\sigma \in \Sigma$ eine Sorte $s \in (S^* \times S)$ zuordnet.[2]

Für $\sigma \in \Sigma$ mit $sort(\sigma) = s$ wird auch σ^s geschrieben. Σ^s bezeichne die Teilmenge von Σ, die alle Symbole σ^s enthält. Ist die Abbildung $sort$ aus dem Kontext ersichtlich, wird für $(\Sigma, sort)$ kurz Σ geschrieben.

□

Definition 2.5 (S-sortierte Bäume)
Sei S eine endliche, nichtleere Menge von Sorten. Sei Σ eine $(S^* \times S)$-sortierte Menge. Sei X eine beliebige S-sortierte Menge. Die *Menge der S-sortierten Bäume über Σ indiziert mit X*, bezeichnet mit $T_\Sigma(X)$, ist die kleinste Menge $\bigcup_{s \in S} T^s$, für die gilt:

- $X^s \subseteq T^s$
- Für alle $\sigma \in \Sigma^{(s_1 \ldots s_n, s_0)}$ mit $n \geq 0$ und $s_0, \ldots, s_n \in S$ und für alle $t_1^{s_1}, \ldots, t_n^{s_n} \in T$ ist $\sigma(t_1, \ldots, t_n) \in T^{s_0}$.

  Für $\sigma(t_1, \ldots, t_n)$ wird auch $\begin{array}{c}\sigma\\ \diagup \quad \diagdown\\ t_1 \ldots t_n\end{array}$ geschrieben.

□

Die Begriffe *Baum* und *Term* werden synonym verwendet.

Definition 2.6 (Operationen über Bäumen)
Sei S eine Menge von Sorten. Sei Σ eine $(S^* \times S)$-sortierte Menge. Sei X eine beliebige S-sortierte Menge. Für jeden Baum $t \in T_\Sigma(X)$ werden definiert:

1. die *Menge der Knoten von t*, bezeichnet mit $node(t)$:

$$\begin{aligned} node(t) = & \{\varepsilon\}, \\ & \text{falls } t \in X \\ = & \{\varepsilon\} \cup \{w | w = iv, i \in \{1, \ldots, k\}, v \in node(t_i)\}, \\ & \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X) \end{aligned}$$

   (Jeder Knoten n wird durch den Pfad von der Wurzel zu n identifiziert.)

2. der *Wurzelknoten von t*, bezeichnet mit $root(t)$:

$$root(t) = \varepsilon$$

[2] Wegen der Endlichkeit der Sortenmenge werden $(S^* \times S)$-sortierte Mengen nicht implizit durch S-sortierte Mengen beschrieben, sondern eine S-sortierte Menge läßt sich als spezielle $(S^* \times S)$-sortierte Menge notieren, bei der der Wertebereich der Abbildung $sort$ auf $(\{\varepsilon\} \times S)$ eingeschränkt wurde.

3. für jeden Knoten $n \in node(t)$ die *Beschriftung von n*, bezeichnet mit $label(t, n)$:

$$
\begin{aligned}
label(t,n) =& t, \\
& \quad \text{falls } t \in X \text{ und } n = \varepsilon, \\
=& \sigma, \\
& \quad \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0 \text{ und } n = \varepsilon \\
=& label(t_i, v), \\
& \quad \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X) \\
& \quad \text{und } n = iv \text{ mit } i \in \{1, \ldots, k\}
\end{aligned}
$$

4. die *Höhe von t*, bezeichnet mit $height(t)$:

$$
\begin{aligned}
height(t) =& 1, \\
& \quad \text{falls } t \in X \\
=& 1 + \max(height(t_1), \ldots, height(t_k)), \\
& \quad \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X)
\end{aligned}
$$

5. die *Größe von t*, bezeichnet mit $size(t)$:

$$size(t) = \mid node(t) \mid$$

(Die Größe eines Baums t entspricht der Anzahl der Knoten von t.)

6. die *Menge der Teilbäume von t*, bezeichnet mit $sub(t)$:

$$
\begin{aligned}
sub(t) =& \{t\}, \\
& \quad \text{falls } t \in X \\
=& \{t\} \cup \bigcup_{i \in \{1, \ldots, k\}} sub(t_i), \\
& \quad \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X)
\end{aligned}
$$

7. die *Menge der Variablen von t*, bezeichnet mit $var(t)$:

$$
\begin{aligned}
var(t) =& \{t\}, \\
& \quad \text{falls } t \in X \\
=& \bigcup_{i \in \{1, \ldots, k\}} var(t_i), \\
& \quad \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X)
\end{aligned}
$$

(In der Menge der Variablen eines Baumes t sind alle Knotenbeschriftungen von t, die Element von X sind.)

8. die *Menge der Symbole von t*, bezeichnet mit $sym(t)$:

$$\begin{aligned} sym(t) =& \emptyset, \\ & \text{falls } t \in X \\ =& \{\sigma\} \cup \bigcup_{i \in \{1,\ldots,k\}} sym(t_i), \\ & \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0, \sigma \in \Sigma^k, t_1, \ldots, t_k \in T_\Sigma(X) \end{aligned}$$

(In der Menge der Symbole eines Baumes t sind alle Knotenbeschriftungen von t, die Element von Σ sind.)

□

Definition 2.7 (Gleichheit zweier Terme)
Zwei Terme $s, t \in T_\Sigma(X)$ heißen genau dann *gleich*, bezeichnet mit $s = t$, wenn gilt:

- $sort(s) = sort(t)$ und
- $node(s) = node(t)$ und
- Für alle $n \in node(s)$ ist $label(s, n) = label(t, n)$.

□

Definition 2.8 (Simultane Termersetzung)
Sei S eine Menge von Sorten. Sei Σ eine $(S^* \times S)$-sortierte Menge. Sei X eine beliebige S-sortierte Menge. Für jeden Term $t \in T_\Sigma(X)$, für jedes $n \in I\!N$, für alle $i \in \{1, \ldots, n\}$, für alle Sorten $s_i \in S$ und für alle Terme $v_i, u_i \in T_\Sigma(X)^{s_i}$ wird die *simultane Termersetzung von Teiltermen von t*, bezeichnet mit $t[v_1 := u_1, \ldots, v_n := u_n]$, wie folgt definiert, wobei für alle $i, j \in \{1, \ldots, n\}$ $v_i \neq v_j$ gilt, falls $i \neq j$ ist:

$$\begin{aligned} t[v_1 := u_1, \ldots, v_n := u_n] =& u_i, \\ & \text{falls } t = v_i \\ =& \sigma(t_1[x_1 := u_1, \ldots, v_n := u_n], \ldots, t_k[v_1 := u_1, \ldots, v_n := u_n]) \\ & \text{falls } t = \sigma(t_1, \ldots, t_k) \text{ mit } k \geq 0 \\ & \text{und für alle } i \in \{1, \ldots, n\} \text{ gilt } t \neq v_i \end{aligned}$$

□

2.4 Kontextfreie Grammatiken

Definition 2.9 (Kontextfreie Grammatik)
Eine *kontextfreie Grammatik G* ist ein Tupel $G = (N, \Sigma, Z, P)$ bestehend aus:

- zwei disjunkten endlichen Mengen N und Σ, wobei die Elemente von N als *Nichtterminalsymbole* und die Elemente von Σ als *Terminalsymbole* bezeichnet werden,

- einem ausgezeichneten Symbol $Z \in N$, bezeichnet als *Startsymbol von* G, und
- einer endlichen Menge $P \subseteq N \times (N \cup \Sigma)^*$ von *Produktionen*. Eine Produktion $p \in P$ wird wie folgt notiert:
$$p : X_0 \to w_0 X_1 w_1 \dots X_n w_n,$$
mit $n \geq 0$ und für alle $i \in \{0, \dots, n\}$ ist $X_i \in N$ und $w_i \in \Sigma^*$. Der Index i heißt *Position des Nichtterminalsymbols* X_i.

□

Definition 2.10 (Ableitungsrelation, ableitbare Symbole)
Sei $G = (N, \Sigma, Z, P)$ eine kontextfreie Grammatik. Die *Ableitungsrelation von* G, bezeichnet mit $\to_G$, ist die kleinste Menge $\bigcup_{p \in P} \to_p$, für die gilt:[3]

Für alle $p : X_0 \to w_0 X_1 w_1 \dots X_n w_n \in P$ und für alle $u, v \in (N \cup \Sigma)^*$ ist

$$(uX_0v, uw_0X_1w_1 \dots X_nw_nv) \in \to_p .$$

Stehen zwei Zeichenketten $v_1, v_2 \in (N \cup \Sigma)^*$ in Ableitungsrelation, so wird dafür auch $v_1 \to_G v_2$ geschrieben.

Für jedes $X_0 \in N$ ist *die Menge der aus* X_0 *ableitbaren Symbole* die Menge

$$\{X \mid X \in (N \cup \Sigma), X_0 {\to_G}^* uXv, u, v \in (N \cup \Sigma)^*\}.$$

□

Definition 2.11 (Sprache einer Grammatik)
Sei $G = (N, \Sigma, Z, P)$ eine kontextfreie Grammatik. Die *Sprache von* G, bezeichnet mit $L(G)$, ist die Menge aller Zeichenketten $w \in \Sigma^*$, für die gilt: $Z {\to_G}^* w$.

Jedes Element $l \in L(G)$ heißt *Satz der Sprache von* G.

□

Eine Ableitung eines Satzes l aus der Sprache einer Grammatik, das heißt die Verkettung derjenigen $\to_p \subseteq \to_G$, die notwendig ist, um zu zeigen, daß tatsächlich $l \in L(G)$ gilt, wird durch einen *Ableitungsbaum* dargestellt.

Definition 2.12 (Ableitungsbaum)
Sei $G = (N, \Sigma, Z, P)$ eine kontextfreie Grammatik. Ein *Ableitungsbaum* t *von* G ist ein geordneter markierter Baum t, dessen Knotenbeschriftung $label_t$ folgenden Bedingungen genügt:[4] Sei $node(t)$ Menge der Knoten von t. Sei $root(t) \in node(t)$ Wurzelknoten von t.

1. $label_t : node(t) \to (N \cup \Sigma \cup \{\varepsilon\})$ ist eine Abbildung der Menge der Knoten des Baums in die Vereinigungsmenge aus der Menge der Nichtterminalsymbole, der Menge der Terminalsymbole und der Menge, die das leere Wort ε enthält.
2. $label_t(root(t)) = Z$.

[3] vergleiche [CES88, Seite 251]
[4] nach [CES88, Seite 8]

3. Ist $x_0 \in node(t)$ Knoten des Baums und ist $label_t(x_0) = X_0$ und sind $x_1, \ldots, x_m$, $m \geq 0$, die m geordneten Nachfolgerknoten von x_0 und ist für jedes $i \in \{1, \ldots, m\}$ $label_t(x_i) = X_i$, dann muß $X_0 \to X_1 \ldots X_m \in P$ gelten.

   Existiert ein $i \in \{1, \ldots, m\}$ mit $label_t(x_i) = \varepsilon$, so ist $m = 1$.

   $prod_t(x_0) = X_0 \to X_1 \ldots X_m$ heißt *die am Knoten x_0 angewendete Produktion.* Die Menge $node(t) \setminus \{x_0, \ldots, x_m\}$ heißt *Kontext von p.*

Die Menge der Ableitungsbäume von G wird mit $DT(G)$ bezeichnet.

□

Der Satz, den ein Ableitungsbaum $t \in DT(G_0)$ repräsentiert, ist gleich der Zeichenkette $yield(t) \in \Sigma^*$, die durch Konkatenation der Markierungen der Blattknoten des Ableitungsbaums von links nach rechts entsteht.

Der Identifikation der Knoten eines Ableitungsbaums t dient die folgende Notation. Sei $x \in node(t)$ ein Knoten von t. Dann bezeichne[5]

- $x.\text{-}1$ den Vorgängerknoten von x, falls $x \neq root(t)$,
- $x.0$ den Knoten selbst und
- $x.i$ den i-ten Nachfolgerknoten von x, falls $x.i$ existiert, das heißt, falls $1 \leq i \leq m$ gilt, wobei m die Anzahl der Nachfolgerknoten von x ist.

Beginnend beim Wurzelknoten läßt sich durch obige Notation jedem Knoten $x \in node(t)$ eindeutig eine Sequenz natürlicher Zahlen zuordnen, die den Weg vom Wurzelknoten über die Nachfolgerrelation zum Knoten x beschreibt. Diese Sequenz wird als *Dewey-Notation* des Knotens x bezeichnet. Zur Vereinfachung werden die Punkte zwischen den Zahlen nicht notiert, wenn kein Knoten existiert, der mehr als 9 Nachfolger besitzt. Die zweite Vereinfachung der Notation ist die Elimination aller Zeichen 0, die in einer Sequenz auftreten und der Gebrauch des leeren Wortes ε anstelle von $root(t)$.

Von der kontextfreien Grammatik G wird angenommen, sie sei *reduziert,* das bedeutet, in der Menge $(N \cup \Sigma)$ ist kein Symbol enthalten, das nicht aus dem Startsymbol Z abgeleitet werden kann, und die Menge N enthält kein Nichtterminalsymbol, das nicht zu einer Zeichenreihe $w \in \Sigma^*$ abgeleitet werden kann.

[5] vergleiche die Menge $node(t)$ aus Definition 2.6

Kapitel 3

Makroattributgrammatiken

Makroattributgrammatiken sind eine Verallgemeinerung der Attributgrammatiken. Die Verallgemeinerung resultiert aus der Einführung von Makroattributen, die Funktionen repräsentieren können.

3.1 Definition

Die Definition der Makroattributgrammatiken basiert auf der Definition, die von Adamski und Vogler in [AV96] angegeben wird. Im Gegensatz zu [AV96] werden jedoch in der vorliegenden Arbeit ausschließlich unkonditionale Makroattributgrammatiken betrachtet.

Definition 3.1 (Makroattributgrammatik)
Eine *Makroattributgrammatik* MAG ist ein Tupel $MAG = (G_0, D, B, R)$bestehend aus:[1]

- einer *kontextfreien Grammatik* G_0,
- einem *semantischen Bereich* D,
- einer *Attributbeschreibung* B und
- einer *Beschreibung* R *semantischer Regeln.*

Die kontextfreie Grammatik $G_0 = (N, \Sigma, Z, P)$ hat die Struktur nach Definition 2.9.

Der semantische Bereich D ist ein Tupel $D = (\mathrm{K}, \Omega, \Phi, \varphi)$bestehend aus:[2]

- einer endlichen Menge K (lies: „kappa“) von Sorten,
- einer K-sortierten Wertmenge Ω,

[1] Die allgemeinere Definition konditionaler Makroattributgrammatiken aus [AV96] definiert mit $MAG = (G_0, D, B, R, C)$ eine weitere Komponente: eine Familie C semantischer Bedingungen, die in der vorliegenden Arbeit nicht betrachtet werden soll.

[2] In [AV96] wird mit $D = (\mathrm{K}, \Omega, \Phi, \Psi, \varphi)$ eine Menge Ψ von Prädikatssymbolen definiert und die Interpretation φ auf die Symbole in Ψ erweitert.

- einer endlichen $(\mathrm{K}^* \times \mathrm{K})$-sortierten Menge Φ von Funktionssymbolen und
- einer Interpretation φ der Funktionssymbole, die jedem Symbol $f \in \Phi$ eine Funktion über Teilmengen von Ω zuordnet:

  Für jedes $k \in I\!N$ und für alle $\kappa_0, \ldots, \kappa_k \in \mathrm{K}$ gilt: Für jedes $f \in \Phi^{(\kappa_1 \ldots \kappa_k, \kappa_0)}$, ist $\varphi(f) : \Omega^{\kappa_1} \times \ldots \times \Omega^{\kappa_k} \to \Omega^{\kappa_0}$, eine k-stellige berechenbare Funktion.

Die Attributbeschreibung B ist ein Tupel $B = (S\!-\!Att, I\!-\!Att, A, \alpha_0)$bestehend aus:[3]

- zwei disjunkten endlichen $(\mathrm{K}^* \times \mathrm{K})$-sortierten Mengen $S\!-\!Att$ und $I\!-\!Att$, wobei die Elemente von $S\!-\!Att$ als *synthetische Attribute* und die Elemente von $I\!-\!Att$ als *inherite Attribute* bezeichnet werden,
- einer Abbildung $A : N \to \mathcal{P}(S\!-\!Att \cup I\!-\!Att)$, die jedem Nichtterminalsymbol $X \in N$ eine Menge von Attributen zuordnet, wobei $A(Z) \cap I\!-\!Att = \emptyset$ angenommen wird.
- einem ausgezeichneten Attribut $\alpha_0 \in A(Z)$ mit $sort(\alpha_0) \in (\{\varepsilon\} \times \mathrm{K})$, das als *Bedeutungsattribut* bezeichnet wird.

Durch die Abbildung A wird jedem Nichtterminalsymbol eindeutig eine Menge von Attributen zugeordnet. Tritt jedoch ein Nichtterminalsymbol X in einer Produktion mehrfach auf, werden die daraus resultierenden verschiedenen Vorkommen der Attribute von X allein durch die Abbildung A *nicht* unterschieden. Jedoch läßt sich jeder Produktion $p : X_0 \to w_0 X_1 w_1 \ldots X_n w_n \in P$, $n \geq 0$, durch Assoziierung der Attribute mit den Indizes $0, \ldots, n$ der Nichtterminalsymbole eindeutig eine $(\mathrm{K}^* \times \mathrm{K})$-sortierte Menge $A(p)$ zuordnen:

$$A(p) = \{\langle a, j\rangle \mid 0 \leq j \leq n, a \in A(X_j)\}.$$

Die Elemente von $A(p)$ heißen *Attributvorkommen von* p, und j heißt der *Index von* $\langle a, j\rangle$. Für alle $\langle a, j\rangle \in A(p)$ gilt $sort(\langle a, j\rangle) = sort(a)$.

Die Menge $A(p)$ läßt sich in zwei disjunkte Teilmengen zerlegen:

- in die Menge der *Innenattributvorkommen von* p:

$$\begin{aligned} innen(p) =& \{\langle a, 0\rangle \mid a \in A(X_0) \cap S\!-\!Att\} \\ & \cup \{\langle a, j\rangle \mid a \in A(X_j) \cap I\!-\!Att, 1 \leq j \leq n\} \end{aligned}$$

- und in die Menge der *Außenattributvorkommen von* p:

$$\begin{aligned} außen(p) =& \{\langle a, 0\rangle \mid a \in A(X_0) \cap I\!-\!Att\} \\ & \cup \{\langle a, j\rangle \mid a \in A(X_j) \cap S\!-\!Att, 1 \leq j \leq n\}. \end{aligned}$$

Zur Illustration von Innen- und Außenattributvorkommen diene die Abbildung 3.1 einer Produktion $p : L \to LL$ und ihrer Attributvorkommen, die aus $A(L) \cap S\!-\!Att = \{w, u\}$ und $A(L) \cap I\!-\!Att = \{v\}$ resultieren.

---

[3] Die Attributbeschreibung ist eine Modifikation des Tupels $B = (S\!-\!Att, I\!-\!Att, Att, S, I, A, \alpha_0, W)$, das in [AV96] angegeben wird: Die Definition der Komponenten $Att = S\!-\!Att \cup I\!-\!Att$, $S : N \to \mathcal{P}(S\!-\!Att)$ und $I : N \to \mathcal{P}(I\!-\!Att)$ ist redundant und kann somit entfallen. Da die Mengen der Attribute sortierte Mengen sind, ist die Komponente W, mit für alle $a \in Att$ ist $W(a) = sort(a)$, redundant und kann somit entfallen.

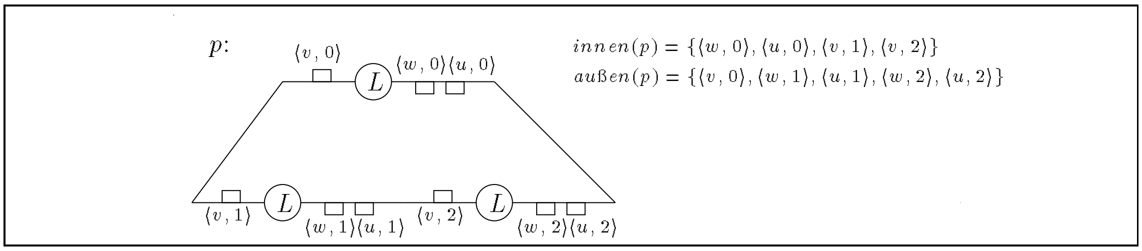


Abbildung 3.1: Illustration von Innen- und Außenattributvorkommen

Die Zerlegung von $A(p)$ bestimmt Form und Anzahl der semantischen Regeln, die zur Spezifikation der Makroattributgrammatik beschrieben werden müssen.

Die Beschreibung R der semantischen Regeln ist ein Tupel $R = (Y, RP)$ bestehend aus:

- einer endlichen K-sortierten Menge Y von Parametern und
- einer Familie $RP = (\ R(p) \mid p \in P\ )$ semantischer Regeln.

Die Familie RP definiert für jede Produktion p eine endliche Menge $R(p)$ von semantischen Regeln, die für jedes Innenattributvorkommen

$$\langle a, j\rangle \in innen(p) \text{ mit } sort(\langle a, j\rangle) = (\kappa_1 \dots \kappa_m, \kappa_0) \text{ und } m \geq 0$$

genau eine semantische Regel der Form

$$\langle a, j\rangle(y_1, \dots, y_m) = \zeta$$

enthält, wobei gilt: für alle $i \in \{1, \dots, m\}$ ist $y_i \in Y^{\kappa_i}$ und $\zeta \in T_{\Phi \cup außen(p)}(\{y_1, \dots, y_m\})^{\kappa_0}$.

□

Die semantischen Regeln einer Produktion p induzieren eine Abhängigkeitsrelation auf den Attributvorkommen von p. Intuitiv ist das Attributvorkommen auf der linken Seite einer semantischen Regel r von all denjenigen Außenattributvorkommen abhängig, die zur Konstruktion des Terms ζ auf der rechten Seite von r benutzt wurden. Um die in ζ auftretenden Attributvorkommen zu bestimmen, muß demzufolge die Struktur von ζ *entschachtelt* werden.

Bei der Betrachtung eines beliebigen $\zeta \in T_{\Phi \cup außen(p)}(Y)$ fällt auf, daß Teilterme von ζ dadurch konstruiert werden können, daß Außenattributvorkommen ineinandergeschachtelt werden. Die Frage, ob dadurch eventuell Abhängigkeiten zwischen den ineinandergeschachtelten Außenattributvorkommen induziert werden, beantwortet Beobachtung 3.2.

Beobachtung 3.2 (Abhängigkeiten verschachtelter Attributvorkommen)
Abhängigkeiten zwischen den Attributvorkommen, die zur Konstruktion einer rechten Seite ζ einer semantischen Regel verschachtelt wurden, lassen sich aus ζ *nicht* ableiten.

□

Beispiel 3.3 (Abhängigkeiten)
In Abbildung 3.2 sind die Abhängigkeiten zwischen den Attributvorkommen $\langle u,0\rangle$, $\langle u,1\rangle$ und $\langle u,2\rangle$ der Produktion $p : L \to LL$ dargestellt, die aus der semantischen Regel $r \in R(p)$ mit

$$r : \langle u,0\rangle(y) = \langle u,1\rangle(\langle u,2\rangle(y))$$

resultieren.

■

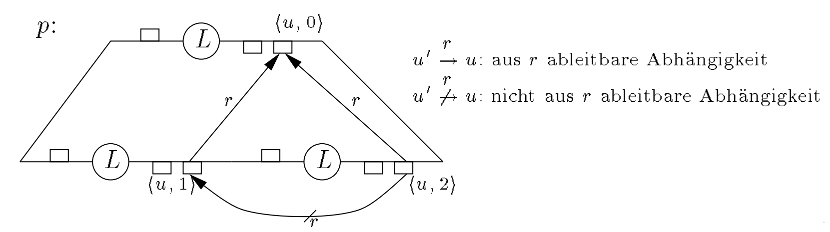


Abbildung 3.2: Illustration von Beobachtung 3.2

Die Beobachtung 3.2 ist leicht einzusehen, wenn die Bedeutung der Terme betrachtet wird. Semantisch stellen Terme der Form $\langle u,1\rangle(\ldots,\langle u,2\rangle(\ldots),\ldots)$ *Funktionsapplikationen* dar, und der Term $\langle u,2\rangle(\ldots)$ repräsentiert eines der *Argumente*, die bei diesem Aufruf der Funktion $\langle u,1\rangle$ übergeben werden. Im Gegensatz zum *Wert*, der durch eine Funktionsapplikation erzeugt wird, und der sowohl von der Funktion als auch von den übergebenen Argumenten abhängig ist, ist die *Funktion* selbst natürlich *nicht* von den aktuellen Argumenten abhängig, die ihr bei einem Aufruf übergeben werden.

Die herrschenden Abhängigkeiten zwischen den Attributvorkommen einer Produktion werden durch den *Abhängigkeitsgraphen einer Produktion* repräsentiert.

Definition 3.4 (Abhängigkeitsgraph einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $p \in P$ eine Produktion der kontextfreien Grammatik G_0. Sei $R(p)$ die Menge der semantischen Regeln von p. Der *Abhängigkeitsgraph der Produktion* p, bezeichnet mit $D(p)$, ist der Graph $D(p) = (A(p), E(p))$ mit der Kantenrelation

$$E(p) = \{(\langle b,j\rangle, \langle a,i\rangle) \mid (\langle a,i\rangle(y_1,\ldots,y_m) = \zeta) \in R(p), \langle b,j\rangle \in (sym(\zeta) \cap A(p))\}.$$

□

3.2 Beispielmakroattributgrammatik

Zur Illustration der eingeführten Konzepte dient eine Makroattributgrammatik, die jedes Wort aus $\{0,1\}^* \setminus \{\varepsilon\}$ als Binärdarstellung einer natürlichen Zahl n interpretiert.

Beispiel 3.5 (Bindec)
$MAG_{10} = (G_0, D, B, R)$

- $G_0 = (N, \Sigma, Z, P)$
  - $N = \{Z, L\}$
  - $\Sigma = \{0, 1\}$
  - $P = \{p_1 : Z \to L,\ p_2 : L \to LL,\ p_3 : L \to 0,\ p_4 : L \to 1\}$
- $D = (\mathrm{K}, \Omega, \Phi, \varphi)$
  - $\mathrm{K} = \{n\}$
  - $\Omega = \Omega^n = I\!N$
  - $\Phi = \{null^{(\varepsilon,n)}, s^{(n,n)}\}$
  - $\varphi(null)() = 0$, und für alle $e \in \Omega^n$ ist $\varphi(s)(e) = e + 1$
- $B = (S\!-\!Att, I\!-\!Att, A, \alpha_0)$
  - $S\!-\!Att = \{\alpha_0^{(\varepsilon,n)}, w^{(n,n)}, u^{(n,n)}\}$
  - $I\!-\!Att = \{v^{(n,n)}\}$
  - $A(Z) = \{\alpha_0\},\ A(L) = \{w, u, v\}$
- $R = (Y, RP)$
  - $Y = \{y^n\}$
  - $RP = (R(p) \mid p \in P)$ mit

$$
\begin{aligned}
R(p_1) = \{ \quad & \langle \alpha_0, 0\rangle = \langle u, 1\rangle(null), \\
& \langle v, 1\rangle(y) = s(y)\} \\
R(p_2) = \{\ & \langle v, 1\rangle(y) = \langle w, 2\rangle(\langle w, 2\rangle(y)), \\
& \langle v, 2\rangle(y) = \langle v, 0\rangle(y), \\
& \langle w, 0\rangle(y) = \langle w, 1\rangle(y), \\
& \langle u, 0\rangle(y) = \langle u, 1\rangle(\langle u, 2\rangle(y))\} \\
R(p_3) = \{ & \langle w, 0\rangle(y) = \langle v, 0\rangle(y), \\
& \langle u, 0\rangle(y) = y\} \\
R(p_4) = \{ & \langle w, 0\rangle(y) = \langle v, 0\rangle(y), \\
& \langle u, 0\rangle(y) = \langle v, 0\rangle(y)\}
\end{aligned}
$$

Die Abhängigkeitsgraphen der Produktionen p_1, p_2, p_3 und p_4 sind in Abbildung 3.3 dargestellt.

■

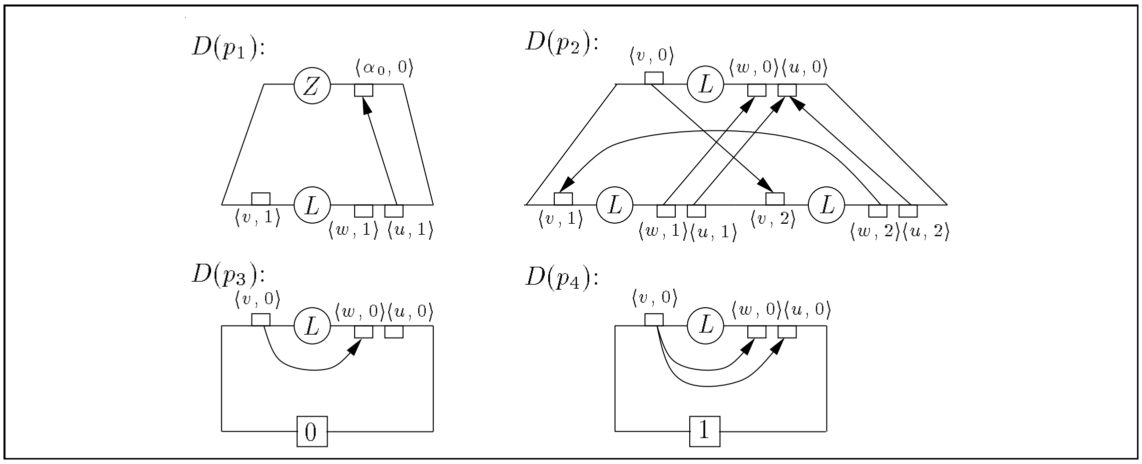


Abbildung 3.3: Abhängigkeitsgraphen der Produktionen p_1, p_2, p_3 und p_4.

3.3 Semantik von Ableitungsbäumen

Durch die Anwendung einer Produktion $p : X_0 \rightarrow w_0X_1w_1 \ldots X_mw_m$ an einem Knoten x werden m neue Knoten des Ableitungsbaums erzeugt, die Instanzen der Nichtterminalsymbole $X_1, \ldots, X_m$. Gleichzeitig ist mit dieser Instanziierung der Nichtterminalsymbole eine Instanziierung der Attribute verbunden, die den Nichtterminalsymbolen $X_1, \ldots, X_m$ zugeordnet sind.

Definition 3.6 (Attributinstanz)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ Ableitungsbaum von G_0. Sei $x \in node(t)$ Knoten von t. Die *Menge der Attributinstanzen von x*, bezeichnet mit $A(x)$, ist die Menge

$$A(x) = \{\langle a, x\rangle \mid a \in A(label_t(x))\}.$$

Die *Menge der Attributinstanzen von t*, bezeichnet mit $A(t)$, ist die Menge

$$A(t) = \bigcup_{x \in node(t)} A(x).$$

Für alle Attributinstanzen $\langle a, x\rangle \in A(t)$ gilt $sort(\langle a, x\rangle) = sort(a)$.

□

Ähnlich den Instanzen der Attribute werden auch die Terme auf den rechten Seiten der semantischen Regeln der Produktion instanziiert, indem sie auf einen Knoten des Ableitungsbaums bezogen werden.

Definition 3.7 (Instanz der rechten Seite)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Sei $x \in node(t)$ ein Knoten von t und $p = prod_t(x)$ die an x angewendete Produktion. Sei $\zeta \in T_{\Phi \cup au\ss en(p)}(Y)^\kappa$, $\kappa \in \mathrm{K}$. Die *Instanz von ζ am Knoten x*, bezeichnet

mit $[\zeta]_x$, ist ein Term aus $T_{\Phi \cup A(t)}(Y)^\kappa$ mit folgender Struktur:

$$
\begin{aligned}
[\zeta]_x = & \zeta, \text{ falls } \zeta \in Y \\
= & f([\zeta_1]_x, \ldots, [\zeta_k]_x), \\
& \quad \text{falls } \zeta = f(\zeta_1, \ldots, \zeta_k) \text{ und } f \in \Phi^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \\
& \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } \zeta_i \in T_{\Phi \cup au\text{ß}en(p)}(Y)^{\kappa_i} \\
= & \langle b, x.j \rangle([\zeta_1]_x, \ldots, [\zeta_k]_x), \\
& \quad \text{falls } \zeta = \langle b, j \rangle(\zeta_1, \ldots, \zeta_k) \text{ und } \langle b, j \rangle \in au\text{ß}en(p)^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \\
& \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } \zeta_i \in T_{\Phi \cup au\text{ß}en(p)}(Y)^{\kappa_i}.
\end{aligned}
$$

□

Eine Instanz der rechten Seite einer semantischen Regel r dient als die rechte Seite einer Instanz von r. Die Instanzen semantischer Regeln heißen *semantische Gleichungen*.

Definition 3.8 (Semantische Gleichung)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ Ableitungsbaum von G_0. Sei $x \in node(t)$, $p = prod_t(x)$. Sei $\langle a, i \rangle(y_1, \ldots, y_m) = \zeta \in R(p)$ eine semantische Regel von p. Dann ist

$$\langle a, x.i \rangle(y_1, \ldots, y_m) = [\zeta]_x$$

eine *semantische Gleichung von t*. Die Menge aller semantischen Gleichungen von t wird mit $R(t)$ bezeichnet.

□

Die semantischen Gleichungen induzieren einen Abhängigkeitsgraphen auf den Attributinstanzen eines Ableitungsbaums.

Definition 3.9 (Abhängigkeitsgraph eines Ableitungsbaums)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Sei $R(t)$ die Menge der semantischen Gleichungen von t. Der *Abhängigkeitsgraph von t*, bezeichnet mit $D(t)$, ist der Graph $D(t) = (A(t), E(t))$ mit der Kantenrelation

$$
\begin{aligned}
E(t) = \{(\langle b, x.j \rangle, \langle a, x.i \rangle) \mid \; & x \in node(t), (\langle a, x.i \rangle(y_1, \ldots, y_m) = [\zeta]_x) \in R(t), \\
& \langle b, x.j \rangle \in (sym([\zeta]_x) \cap A(t)) \; \}
\end{aligned}
$$

□

Beispiel 3.10 (Fortsetzung von 3.5)
Ein Satz aus der Sprache der Beispielgrammatik MAG_{10} aus Abschnitt 3.2 ist die Zeichenkette $w = 1010$. Ein Ableitungsbaum für w ist der in Abbildung 3.4 dargestellte Ableitungsbaum t_{10}. Der Abhängigkeitsgraph $D(t_{10})$ resultiert aus der Menge $R(t_{10})$, die folgende semantischen Gleichungen enthält:

$$
\begin{array}{rclrcl}
\langle v,1\rangle(y) &=& s(y) & \langle v,12\rangle(y) &=& \langle v,1\rangle(y)\\
\langle v,122\rangle(y) &=& \langle v,12\rangle(y) & \langle w,122\rangle(y) &=& \langle v,122\rangle(y)\\
\langle u,122\rangle(y) &=& y & \langle v,121\rangle(y) &=& \langle w,122\rangle(\langle w,122\rangle(y))\\
\langle v,1212\rangle(y) &=& \langle v,121\rangle(y) & \langle w,1212\rangle(y) &=& \langle v,1212\rangle(y)\\
\langle u,1212\rangle(y) &=& \langle v,1212\rangle(y) & \langle v,1211\rangle(y) &=& \langle w,1212\rangle(\langle w,1212\rangle(y))\\
\langle w,1211\rangle(y) &=& \langle v,1211\rangle(y) & \langle u,1211\rangle(y) &=& y\\
\langle w,121\rangle(y) &=& \langle w,1211\rangle(y) & \langle u,121\rangle(y) &=& \langle u,1211\rangle(\langle u,1212\rangle(y))\\
\langle w,12\rangle(y) &=& \langle w,121\rangle(y) & \langle u,12\rangle(y) &=& \langle u,121\rangle(\langle u,122\rangle(y))\\
\langle v,11\rangle(y) &=& \langle w,12\rangle(\langle w,12\rangle(y)) & \langle w,11\rangle(y) &=& \langle v,11\rangle(y)\\
\langle u,11\rangle(y) &=& \langle v,11\rangle(y) & \langle w,1\rangle(y) &=& \langle w,11\rangle(y)\\
\langle u,1\rangle(y) &=& \langle u,11\rangle(\langle u,12\rangle(y)) & \langle \alpha_0,\varepsilon\rangle &=& \langle u,1\rangle(null).
\end{array}
$$

■

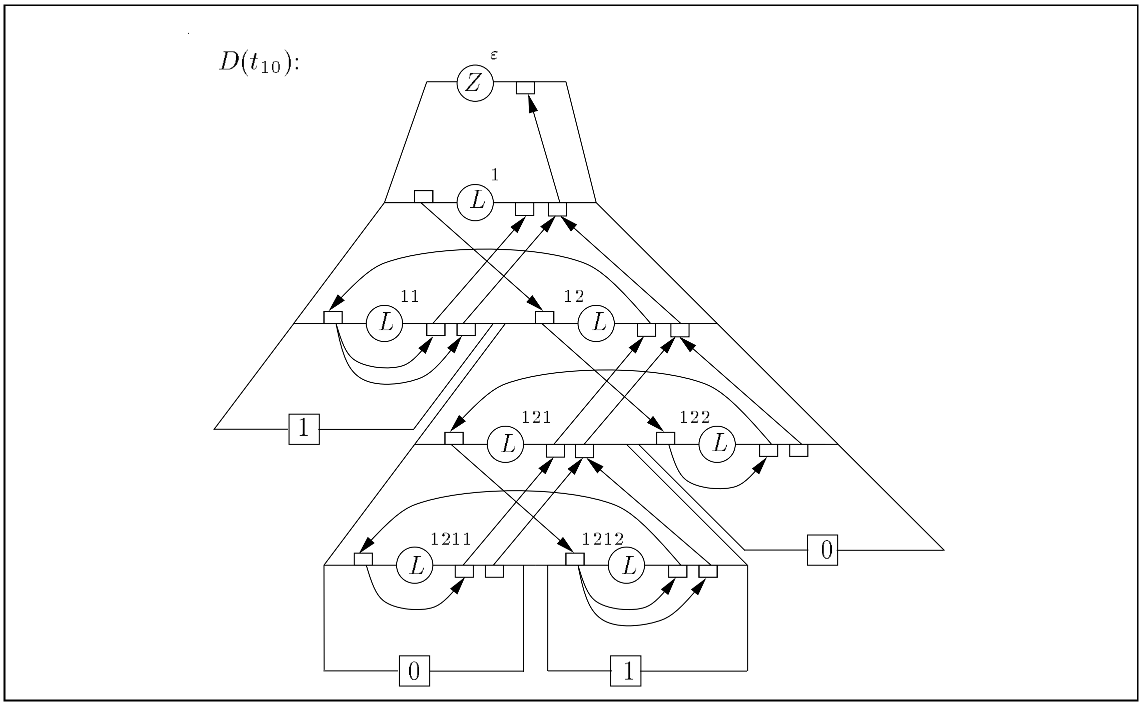


Abbildung 3.4: Abhängigkeitsgraph des Ableitungsbaums t_{10} des Satzes 1010

Definition 3.11 (Dekoration eines Ableitungsbaums)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Eine *Dekoration von t*, bezeichnet mit val_t, ist eine Abbildung

$$val_t : A(t) \to Fun(\Omega),$$

wobei $Fun(\Omega) = \{f \mid f : \Omega^{\kappa_1} \times \ldots \times \Omega^{\kappa_k} \to \Omega^{\kappa_0}, k \geq 0, \kappa_0, \ldots, \kappa_k \in \mathrm{K}\}$, sodaß für alle $\langle a, x\rangle^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \in A(t)$ gilt:
$val_t(\langle a, x\rangle)$ ist eine berechenbare Funktion des Typs $\Omega^{\kappa_1} \times \ldots \times \Omega^{\kappa_k} \to \Omega^{\kappa_0}$.

□

Aus der Menge der möglichen Dekorationen eines Ableitungsbaums sind diejenigen von Bedeutung, die die semantischen Gleichungen des Ableitungsbaums erfüllen.

Definition 3.12 (Zulässige Dekoration)
Eine Dekoration val_t heißt genau dann *zulässig*, wenn für alle semantischen Gleichungen

$$\langle a, x.i\rangle(y_1, \ldots, y_m) = [\zeta]_x$$

mit $sort(\langle a, x.i\rangle) = (\kappa_1 \ldots \kappa_m, \kappa_0)$ gilt: Für alle $c_i \in \Omega^{\kappa_i}$ mit $i \in \{1, \ldots, m\}$ ist

$$val_t(\langle a, x.i\rangle)(c_1, \ldots, c_m) = \Theta([\zeta]_x, c_1, \ldots, c_m),$$

wobei $\Theta([\zeta]_x, c_1, \ldots, c_m)$ folgendermaßen definiert wird:

$$\begin{aligned}
\Theta([\zeta]_x, c_1, \ldots, c_m) = & c_i, \text{ falls } [\zeta]_x = y_i \text{ und } y_i \in Y \\
= & \varphi(f)(\Theta([\zeta_1]_x, c_1, \ldots, c_m), \ldots, \Theta([\zeta_k]_x, c_1, \ldots, c_m)), \\
& \quad \text{falls } [\zeta]_x = f([\zeta_1]_x, \ldots, [\zeta_k]_x) \text{ und } f \in \Phi^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \\
& \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \\
& \quad \text{ist } \kappa_i \in \mathrm{K} \text{ und } [\zeta_i]_x \in T_{\Phi \cup A(t)}(Y)^{\kappa_i} \\
= & val_t(\langle b, x.j\rangle)(\Theta([\zeta_1]_x, c_1, \ldots, c_m), \ldots, \Theta([\zeta_k]_x, c_1, \ldots, c_m)) \\
& \quad \text{falls } [\zeta]_x = \langle b, x.j\rangle([\zeta_1]_x, \ldots, [\zeta_k]_x) \text{ und } \langle b, x.j\rangle \in A(t)^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \\
& \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \\
& \quad \text{ist } \kappa_i \in \mathrm{K} \text{ und } [\zeta_i]_x \in T_{\Phi \cup A(t)}(Y)^{\kappa_i}.
\end{aligned}$$

□

Beispiel 3.13 (Fortsetzung von 3.5)
Die folgende Dekoration $val_{t_{10}}$ des Ableitungsbaums t_{10} aus Abbildung 3.4 ist zulässig. Für alle $y \in I\!N$ gilt:

$$\begin{array}{lll}
val_{t_{10}}(\langle v, 1\rangle)(y) = y+1 & val_{t_{10}}(\langle v, 12\rangle)(y) = y+1 & val_{t_{10}}(\langle v, 122\rangle)(y) = y+1 \\
val_{t_{10}}(\langle w, 122\rangle)(y) = y+1 & val_{t_{10}}(\langle u, 122\rangle)(y) = y & val_{t_{10}}(\langle v, 121\rangle)(y) = y+2 \\
val_{t_{10}}(\langle v, 1212\rangle)(y) = y+2 & val_{t_{10}}(\langle w, 1212\rangle)(y) = y+2 & val_{t_{10}}(\langle u, 1212\rangle)(y) = y+2 \\
val_{t_{10}}(\langle v, 1211\rangle)(y) = y+4 & val_{t_{10}}(\langle w, 1211\rangle)(y) = y+4 & val_{t_{10}}(\langle u, 1211\rangle)(y) = y \\
val_{t_{10}}(\langle w, 121\rangle)(y) = y+4 & val_{t_{10}}(\langle u, 121\rangle)(y) = y+2 & val_{t_{10}}(\langle w, 12\rangle)(y) = y+4 \\
val_{t_{10}}(\langle u, 12\rangle)(y) = y+2 & val_{t_{10}}(\langle v, 11\rangle)(y) = y+8 & val_{t_{10}}(\langle w, 11\rangle)(y) = y+8 \\
val_{t_{10}}(\langle u, 11\rangle)(y) = y+8 & val_{t_{10}}(\langle w, 1\rangle)(y) = y+8 & val_{t_{10}}(\langle u, 1\rangle)(y) = y+10 \\
val_{t_{10}}(\langle \alpha_0, \varepsilon\rangle) = 10 & &
\end{array}$$

■

Bei gegebener zulässiger Dekoration jedes Ableitungsbaums führt die alleinige Betrachtung des Wertes der Instanz des Bedeutungsattributs α_0 des jeweiligen Ableitungsbaums zum Begriff der *Übersetzung eines Ableitungsbaums in einen Wert*, die von einer Makroattributgrammatik spezifiziert wird.

Definition 3.14 (Übersetzung einer Makroattributgrammatik)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik.[4] Die von MAG beschriebene *Übersetzung*, bezeichnet mit $\tau_{tv}(MAG)$ (tv lies: „tree to value“), ist die Relation

$$\tau_{tv}(MAG) = \{(t, val_t(\langle \alpha_0, \varepsilon \rangle)) \mid t \in DT(G_0) \text{ und } val_t \text{ ist zulässige Dekoration von } t\}.$$

□

Beispiel 3.15 (Fortsetzung von 3.5)
Mit $val_{t_{10}}(\langle \alpha_0, \varepsilon \rangle) = 10$ gilt: $(t_{10}, 10) \in \tau_{tv}(MAG_{10})$.

■

Sowohl die zulässige Dekoration als auch die Übersetzungsrelation beschreiben eine Abbildung jedes Ableitungsbaums in den semantischen Bereich einer Makroattributgrammatik, wobei die Existenz der Übersetzung per Definition an die Existenz einer zulässigen Dekoration gekoppelt ist.

Im Zusammenhang mit der Frage nach der Existenz einer zulässigen Dekoration eines Ableitungsbaums t wirft die zur Definition der zulässigen Dekoration benutzte Gleichung

$$val_t(\langle a, x.i \rangle)(c_1, \ldots, c_m) \quad = \quad \Theta([\zeta]_x, c_1, \ldots, c_m) \tag{3.1}$$

ein Entscheidbarkeitsproblem auf. Durch Gleichung 3.1 wird, falls $m > 0$ ist, die Werteverlaufsgleichheit zweier Funktionen gefordert: die Funktion $val_t(\langle a, x.i \rangle)$ soll für alle möglichen m-Tupel von Parametern denselben Wert wie die Funktion $\Theta([\zeta]_x)$ liefern. Diese Gleichheit ist ein unentscheidbares Problem.

Das Unentscheidbarkeitsproblem läßt sich jedoch umgehen, wenn berücksichtigt wird, daß es bei der Definition der Semantik eines Ableitungsbaums *nicht* um die Bestimmung der Menge *aller* zulässigen Dekorationen eines Ableitungsbaums geht, sondern um die Bestimmung einer einzigen zulässigen Dekoration, zumal wegen Definition 3.12 je zwei zulässige Dekorationen aus der Menge aller zulässigen Dekorationen eines Ableitungsbaums äquivalent sind. Mit dieser Überlegung wird das Problem der Existenz einer zulässigen Dekoration auf das Problem der *Konstruierbarkeit* einer zulässigen Dekoration reduziert.

Die Konstruktion ist jedoch nicht mittels

> **repeat**
>   Rate eine Dekoration val_t!
> **until** val_t ist zulässig

realisierbar, da der Test der Schleifenbedingung nicht terminieren muß.

[4] Die Definition 3.14 weicht stark von der Definition in [AV96] ab: In [AV96] werden Übersetzungen ausschließlich für nichtzirkuläre Makroattributgrammatiken erklärt. In der vorliegenden Arbeit werden die nichtzirkulären Makroattributgrammatiken aus methodischen Überlegungen erst mit Definition 3.16 eingeführt.

Einen Ansatzpunkt für die Konstruierbarkeit einer zulässigen Dekoration, bietet die Form der semantischen Gleichungen, wobei für alle Attributinstanzen die Gültigkeit der Gleichung 3.1 garantiert und so die Notwendigkeit der Durchführung des problematischen Tests beseitigt ist:

Die linke Seite jeder semantischen Gleichung wird durch genau eine Attributinstanz $\langle a, x.i\rangle$ gebildet, und auf der rechten Seite jeder semantischen Gleichung steht ein Term $[\zeta]_x$ über Funktionssymbolen und Attributinstanzen indiziert mit Parametern.

Unter der Annahme, daß für jede Attributinstanz, die in $[\zeta]_x$ vorkommt, eindeutig ein Wert bestimmt ist, ist $\Theta([\zeta]_x)$ ebenfalls eindeutig bestimmt. Dann bezeichnet $\Theta([\zeta]_x)$ genau eine Äquivalenzklasse bezüglich der Gleichheitsrelation. Anstatt einen Wert $val_t(\langle a, x.i\rangle)$ zu raten und zu testen, ob er und $\Theta([\zeta]_x)$ zu derselben Äquivalenzklasse gehören, wird $\Theta([\zeta]_x)$ selbst als der Wert von $\langle a, x.i\rangle$ eingesetzt. Damit ist die Gleichheit von $val_t(\langle a, x.i\rangle)$ und $\Theta([\zeta]_x)$ garantiert.

Voraussetzung für die Verwendbarkeit der semantischen Gleichungen als „semantische Zuweisungsgleichungen“ ist die Eindeutigkeit der rechten Seiten der Gleichungen.

Adamski und Vogler zeigen in [AV96], daß ein hinreichendes Kriterium für die Eindeutigkeit der rechten Seiten der semantischen Gleichungen und damit für die Existenz einer zulässigen Dekoration die Nichtzirkularität einer Makroattributgrammatik ist.

Definition 3.16 (Nichtzirkuläre Makroattributgrammatik)
Eine Makroattributgrammatik $MAG = (G_0, D, B, R)$ heißt genau dann *nichtzirkulär*, wenn für jeden Ableitungsbaum $t \in DT(G_0)$ gilt, daß sein Abhängigkeitsgraph $D(t)$ ein azyklischer Graph ist.

Die Klasse der nichtzirkulären Makroattributgrammatiken wird mit $NZMAG$ bezeichnet.

□

Ein Algorithmus für den Zirkularitätstest von Makroattributgrammatiken wird in [AV96] diskutiert. Der Test basiert auf der Analyse der Abhängigkeitsgraphen der Produktionen einer gegebenen Grammatik und läßt sich auf den Zirkularitätstest für Attributgrammatiken zurückführen.

Die Berechnung der zulässigen Dekoration eines Ableitungsbaums, respektive der Übersetzung, die durch eine nichtzirkuläre Makroattributgrammatik beschrieben wird, wird mit Algorithmen realisiert, die als *Attributauswerter* bezeichnet werden.

3.4 Attributauswerter

Definition 3.17 (Attributauswerter)
Sei $MAG = (G_0, D, B, R)$ eine nichtzirkuläre Makroattributgrammatik. Ein *Attributauswerter für MAG* ist ein Algorithmus, der für jeden Ableitungsbaum t von G_0 die zulässige Dekoration val_t von t berechnet.

□

Intuitiv besteht die Aufgabe eines Attributauswerters im Auffinden der Lösung des Gleichungssystems $R(t)$, bestehend aus allen semantischen Gleichungen eines Ableitungsbaums t. Das Gleichungssystem $R(t)$ hat folgende Eigenschaften:

1. Die Anzahl der zu berechnenden Attributinstanzen entspricht der Anzahl der Gleichungen.

2. Jede Gleichung ist bereits nach einer Attributinstanz aufgelöst.

3. Das Gleichungssystem ist nichtzirkulär, daß heißt, keine der Attributinstanzen ist von sich selbst abhängig.

Die Berechnung des Wertes der Attributinstanz $\langle a, x.i\rangle$ auf der linken Seite einer semantischen Gleichung

$$\langle a, x.i\rangle(y_1, \dots, y_m) = [\zeta]_x$$

erfolgt durch Reduktion des Terms $[\zeta]_x$ auf der rechten Seite der Gleichung und Zuweisung an $val_t(\langle a, x.i\rangle)$. Zu beachten ist hierbei, daß $[\zeta]_x$ einen Funktionsterm mit den Parametern $y_1, \dots, y_m$, $m \geq 0$, darstellt.

Die Reihenfolge, in der die semantischen Gleichungen ausgewertet werden, muß der Halbordnung genügen, die durch den Abhängigkeitsgraphen des Ableitungsbaums induziert wird.

Die Abhängigkeitsgraphen der Produktionen von Makroattributgrammatiken weisen die gleiche Struktur auf, wie die Abhängigkeitsgraphen der Produktionen von Attributgrammatiken, obwohl die Terme auf den rechten Seiten der semantischen Gleichungen von Makroattributgrammatiken komplexere Strukturen bilden können. Wegen Beobachtung 3.2 führen die komplexeren Termstrukturen jedoch nicht zu neuartigen oder verallgemeinerten Abhängigkeitsgraphen. Mit folgender Konsequenz für die Existenz von Auswertern für Makroattributgrammatiken:

Beobachtung 3.18 (Auswerter für Makroattributgrammtaiken)
Wegen der Übereinstimmung der Struktur der Abhängigkeitsgraphen der Produktionen von Makroattributgrammatiken mit der Struktur der Abhängigkeitsgraphen der Produktionen von Attributgrammatiken sind sämtliche Auswertungsstrategien für Attributgrammatiken, die auf Eigenschaften der Abhängigkeitsgraphen der Produktionen basieren, für Makroattributgrammatiken ebenfalls anwendbar.

□

Demnach ist eine Reihe von Auswertern für Makroattributgrammatiken durch Adaption der existierenden Auswertungsstrategien für Attributgrammatiken konstruierbar, das heißt durch Modifikation der Auswerter, die die Strategien implementieren.

Eine Übersicht zu Auswertungsstrategien für Attributgrammatiken ist beispielsweise Bestandteil von [Vog95].

In der Arbeit von Adamski und Vogler [AV96] werden zwei der bekannten Auswertungsstrategien als mögliche Strategien für Makroattributgrammatiken untersucht: eine bedarfsgesteuerte Strategie, für die ein Auswerter mit der Bezeichnung *Ad Eva* (lies: „Argument driven Evaluator") angegeben wird, und eine baumtraversierende Strategie, implementierbar durch einen Algorithmus *Pv Eva* (lies: „Pure visit Evaluator").

Bedarfsgesteuerte Auswerter

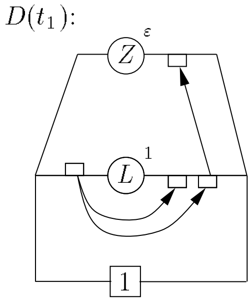
Bedarfsgesteuerte Auswerter basieren auf der Annahme, daß ausschließlich die Übersetzung, die von einer Makroattributgrammatik beschrieben wird, berechnet werden soll. Dazu muß der Auswerter ausschließlich den Wert $val_t(\langle \alpha_0, \varepsilon \rangle)$ der Instanz des Bedeutungsattributs bestimmen.

Beginnend bei der Instanz des Bedeutungsattributs realisiert ein bedarfsgesteuerter Auswerter einen rekursiven Abstieg über den Abhängigkeitsgraphen *je nach Bedarf*. Das heißt, daß bei der Berechnung des Wertes des Terms auf der rechten Seite einer semantischen Gleichung möglicherweise die Werte von Attributinstanzen benötigt werden, denen noch kein Wert zugewiesen wurde. Dann wird die Auswertung der semantischen Gleichung aufgeschoben und erstmal die Werte aller benötigten Attributinstanzen berechnet.

Zwangsläufig hat diese Methode zur Folge, daß die Werte derjenigen Attributinstanzen, die nicht zur Berechnung des Wertes der Instanz des Bedeutungsattributs beitragen, auch nicht berechnet werden. Durch diese Art der Eliminierung überflüssiger Berechnungen spart ein bedarfsgesteuerter Auswerter natürlich Zeit gegenüber einem Auswerter derselben Makroattributgrammatik, der den Wert *jeder* Attributinstanz eines Ableitungsbaums bestimmt.

Aus der Definition 3.12 der zulässigen Dekoration kann ein solcher bedarfsgesteuerter Auswerter gewonnen werden: Die Θ-Reduktion beschreibt eine rekurrente Entwicklung des Wertes der rechten Seite einer semantischen Gleichung, und jede korrekte Implementation der Θ-Reduktion ist ein bedarfsgesteuerter Auswerter.

Beispiel 3.19 (Bedarfsgesteuerte Auswertung)
Zur Illustration der Funktionsweise eines bedarfsgesteuerten Auswerters, der die Θ-Reduktion implementiert, diene der Ableitungsbaum $t_1 \in DT(G_0)$ mit der Menge $R(t_1)$ semantischer Gleichungen bezüglich M_{10}:



$$
\begin{aligned}
R(t_1) = \{ \quad \langle \alpha_0, \varepsilon \rangle &= \langle u, 1 \rangle (null), \\
\langle u, 1 \rangle (y) &= \langle v, 1 \rangle (y) \\
\langle w, 1 \rangle (y) &= \langle v, 1 \rangle (y) \\
\langle v, 1 \rangle (y) &= s(y) \}
\end{aligned}
$$

Die schrittweise Anwendung der Θ-Reduktion erzeugt folgende Terme als Zwischenergebnisse:

$$
\begin{aligned}
val_{t_1}(\langle\alpha_0,\varepsilon\rangle) &= \Theta(\langle u,1\rangle(null))\\
&= val_{t_1}(\langle u,1\rangle)(\Theta(null))\\
&= val_{t_1}(\langle u,1\rangle)(\varphi(null)())\\
&= val_{t_1}(\langle u,1\rangle)(0)\\
&= \Theta(\langle v,1\rangle(y),0)\\
&= val_{t_1}(\langle v,1\rangle)(\Theta(y,0))\\
&= val_{t_1}(\langle v,1\rangle)(0)\\
&= \Theta(s(y),0)\\
&= \varphi(s)(\Theta(y,0))\\
&= \varphi(s)(0)\\
&= 1
\end{aligned}
$$

Damit ist das Tupel $(t_1, 1)$ ein Element in $\tau_{tv}(MAG_{10})$. Die Berechnung von $val_{t_1}(\langle w,1\rangle)$ wird nicht ausgeführt, da sie keinen Beitrag zur Berechung von $val_t(\langle\alpha_0,\varepsilon\rangle)$ leistet.

■

Der Idee, die Θ-Reduktion aus der Definition zu verwenden, kommt die Sortierung des Bedeutungsattributs entgegen. Wegen $sort(\alpha_0) = (\varepsilon, \kappa)$, $\kappa \in \mathrm{K}$, hat die semantische Gleichung zur Bestimmung von $\langle\alpha_0,\varepsilon\rangle$ stets folgende Form:

$$\langle\alpha_0,\varepsilon\rangle = [\zeta]_\varepsilon.$$

Bei der Auswertung gemäß Θ-Reduktion

$$val_t(\langle\alpha_0,\varepsilon\rangle) = \Theta([\zeta]_\varepsilon)$$

wird die rechte Seite *niemals* zu einer parametrisierten Funktion sondern stets zu einer Konstanten ausgewertet werden. Diese Eigenschaft setzt sich bei der rekursiven Anwendung von Θ auf die Teilterme von $[\zeta]_\varepsilon$ fort, was von entscheidender Bedeutung für die Anwendbarkeit der Θ-Reduktion ist.

Während der Rechnung gemäß Θ müssen Funktionssymbole interpretiert werden. Eine Interpretation ist jedoch nur zulässig, falls alle Argumente eines Funktionsaufrufs zur Verfügung stehen. Da $val_t(\langle\alpha_0,\varepsilon\rangle)$ ein „atomarer" Wert aus Ω^κ ist, muß jede Funktionsanwendung, die durch $\Theta([\zeta]_\varepsilon)$ erzeugt wird, einen „atomaren", nicht funktionalen Wert liefern und damit vollständig in allen Parameterpositionen besetzt sein.

Dies gilt nicht für jede Auswertungsstrategie, sodaß die Verwendung der Θ-Reduktion als Auswerter auf eine bedarfsgesteuerte Strategie beschränkt bleibt.

Weitere Eigenschaften der bedarfsgesteuerten Strategie werden nebst eines kompletten Algorithmus zur bedarfsgesteuerten Auswertung in [AV96] diskutiert, sodaß hier nicht weiter darauf eingegangen werden soll.

Traversierende Auswerter

Ein traversierender Auswerter berechnet die zulässige Dekoration eines Ableitungsbaums, indem er den Ableitungsbaum durchläuft und bei jedem Knotenwechsel eine Teilmenge der Attributinstanzen des erreichten Knotens mit Werten belegt. Nach jedem ausgeführten Schritt muß ein traversierender Auswerter zwei Entscheidungen erneut treffen. Die erste Entscheidung betrifft den Wechsel zwischen zwei Knoten des Ableitungsbaums, um die Traversierung fortzusetzen. Die zweite Entscheidung betrifft die Auswahl von Attributinstanzen, deren Werte als nächstes zu berechnen sind.

Zur Steuerung der Traversierung hat der Auswerter die Wahl zwischen genau drei Möglichkeiten: Wechsel zum Vorgängerknoten oder Wechsel zu einem Nachfolgerknoten oder Weiterarbeiten ohne Knotenwechsel. Besucht der Auswerter augenblicklich den Knoten x und ist ihm die Produktion bekannt, die am Knoten x angewendet wurde, steht ihm für die Berechnung jede Attributinstanz $\langle a, x.j\rangle$ zur Auswahl, die aus der Instanziierung eines Innenattributvorkommens $\langle a, j\rangle \in innen(p)$ resultiert.

Das Konzept der traversierenden Auswerter ist nicht neu und für Attributgrammatiken seit geraumer Zeit bekannt. Eine nichtdeterministische Variante eines solchen Auswerters für Makroattributgrammatiken wurde erstmals in [AV96] vorgestellt: Der *Pure-Visit-Evaluator*, der jeden Schritt „raten“ muß.

Im Abschnitt 3.4 der vorliegenden Arbeit wurde die Behauptung aufgestellt, daß sich die Θ-Reduktion höchstens für die Auswertung nach einem bedarfsgesteuerten Schema eignet. Die Begründung hierfür lautet: Die Θ-Reduktion rechnet im semantischen Bereich und „reduziert“ Terme über Funktionssymbolen $f \in \Phi$ durch deren Interpretation $\varphi(f)$ und Anwendung auf ein passendes Tupel von Argumenten. Während der Traversierung eines Ableitungsbaums wird der Wert einer Attributinstanz jedoch beginnend bei den „Blättern“ des Abhängigkeitsgraphen berechnet, das heißt, zuerst wird eine Menge von k Teiltermen berechnet und anschließend mit dem Wurzelterm verknüpft. Bei dieser Berechnungsreihenfolge ist es nicht zulässig, eine Interpretation $\varphi(f)$ eines Funktionssymbols $f \in \Phi$ vorzunehmen, da die Parameter eines Funktionsaufrufs noch nicht bestimmt sind.

Nicht zuletzt wird eine Repräsentation für die berechneten Funktionen benötigt, die die spätere Benutzung bei der Berechnung der Werte anderer Attributinstanzen erlaubt.

Beispiel 3.20 (Kritik der Θ-Reduktion)
Wird im Gleichungssystem aus Beispiel 3.19 versucht, $val_{t_1}(\langle v, 1\rangle)$ unabhängig vom übergebenen Argument zu bestimmen, stößt die Anwendung der Θ-Reduktion auf folgendes Problem:

$$\begin{aligned} val_{t_1}(\langle v, 1\rangle) &= \Theta(s(y)) \\ &= \varphi(s)(\Theta(y)) \end{aligned}$$

Welchen Wert hat $\Theta(y)$? Welche Struktur hat $\varphi(s)$? Intuitiv wird eine Repräsentation für $val_{t_1}(\langle v, 1\rangle)$ erwartet, die auch bei der späteren Benutzung zur Berechnung des Wertes von $\langle \alpha_0, \varepsilon\rangle$ eingesetzt werden kann. Stattdessen ist $val_{t_1}(\langle v, 1\rangle)$ unbestimmt, denn der Ansatz, für alle $c \in I\!N$ gilt:

$$\begin{aligned} val_{t_1}(\langle v,1\rangle)(c) =& \Theta(s(y),c) \\ =& \varphi(s)(\Theta(y,c)) \\ =& \varphi(s)(c) \\ =& c+1, \end{aligned}$$

kann *nicht* als Lösung aufgefaßt werden, da er zu einer Repräsentation der Funktion $val_{t_1}(\langle v,1\rangle)$ in Form einer Relation $\{(c,c+1)\} \subset I\!N \times I\!N$ führt, die unendlich viele Tupel enthält.

■

Eine Möglichkeit zur Repräsentation des Wertes einer Attributinstanz $\langle a, x.i\rangle^{(\kappa_1\ldots\kappa_m,\kappa_0)}$ mit $m \geq 0$ und $\kappa_0,\ldots,\kappa_m \in \mathrm{K}$ ist die Berechnung eines *Terms*, der die Funktion $val_t(\langle a,x.i\rangle)$ *beschreibt*. Ein solcher Term kann aus der rechten Seite $[\zeta]_x$ der semantischen Gleichung $\langle a,x.i\rangle(y_1,\ldots,y_m) = [\zeta]_x$ gewonnen werden. Für die Berechnung des Funktionsterms darf der Auswerter jedoch den Bereich der Terme $T_\Phi(Y)$ über Funktionssymbolen aus Φ und Parametern aus Y nicht verlassen:

Definition 3.21 (Termdekoration eines Ableitungsbaums)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Eine *Termdekoration von* t, bezeichnet mit $wert_t$, ist eine Abbildung

$$wert_t : A(t) \to T_\Phi(Y)$$

wobei für alle $\langle a,x\rangle^{(\kappa_1\ldots\kappa_k,\kappa_0)} \in A(t)$ gilt: $wert_t(\langle a,x\rangle) \in T_\Phi(Y)^{\kappa_0}$ und $var(wert_t(\langle a,x\rangle)) \subseteq \{y_1^{\kappa_1},\ldots,y_k^{\kappa_k}\}$.

□

Definition 3.22 (Zulässige Termdekoration)
Eine Termdekoration $wert_t$ heißt genau dann *zulässig*, wenn für alle semantischen Gleichungen

$$\langle a,x.i\rangle(y_1,\ldots,y_m) = [\zeta]_x,$$

mit $sort(\langle a,x.i\rangle) = (\kappa_1\ldots\kappa_m,\kappa_0)$ gilt:

$$wert_t(\langle a,x.i\rangle) = \Xi([\zeta]_x),$$

wobei die Abbildung $\Xi : T_{\Phi\cup A(t)}(Y) \to T_\Phi(Y)$ folgendermaßen definiert ist: Für alle $z \in T_{\Phi\cup A(t)}(Y)$ ist

$$\begin{aligned} \Xi(z) =& z, \text{ falls } z \in Y \\ =& f(\Xi(z_1),\ldots,\Xi(z_k)), \\ & \quad \text{falls } z = f(z_1,\ldots,z_k) \text{ und } f \in \Phi^{(\kappa_1\ldots\kappa_k,\kappa_0)} \\ & \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1,\ldots,k\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } z_i \in T_{\Phi\cup A(t)}(Y)^{\kappa_i} \\ =& wert_t(\langle b,x.j\rangle)[y_1 := \Xi(z_1),\ldots,y_k := \Xi(z_k)], \\ & \quad \text{falls } z = \langle b,x.j\rangle(z_1,\ldots,z_k) \text{ und } \langle b,x.j\rangle \in A(t)^{(\kappa_1\ldots\kappa_k,\kappa_0)} \\ & \quad \text{mit } k \geq 0 \text{ und für alle } i \in \{1,\ldots,k\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } z_i \in T_{\Phi\cup A(t)}(Y)^{\kappa_i} \end{aligned}$$

□

Beispiel 3.23 (Fortsetzung von 3.19)
Die Anwendung der Ξ-Reduktion zur Bestimmung der zulässigen Termdekoration des Ableitungsbaums t_1 liefert

$$\begin{aligned}
wert_{t_1}(\langle v,1\rangle) &= \Xi(s(y))\\
&= s(\Xi(y))\\
&= s(y)\\
wert_{t_1}(\langle u,1\rangle) &= \Xi(\langle v,1\rangle(y))\\
&= wert_{t_1}(\langle v,1\rangle)[y := \Xi(y)]\\
&= s(y)[y := \Xi(y)]\\
&= s(y)[y := y]\\
&= s(y)\\
wert_{t_1}(\langle w,1\rangle) &= \Xi(\langle v,1\rangle(y))\\
&= wert_{t_1}(\langle v,1\rangle)[y := \Xi(y)]\\
&= s(y)[y := \Xi(y)]\\
&= s(y)[y := y]\\
&= s(y)\\
wert_{t_1}(\langle \alpha_0,\varepsilon\rangle) &= \Xi(\langle u,1\rangle(null))\\
&= wert_{t_1}(\langle u,1\rangle)[y := \Xi(null)]\\
&= s(y)[y := \Xi(null)]\\
&= s(y)[y := null]\\
&= s(null)
\end{aligned}$$

Da zu Beginn der Berechnung des Wertes der Attributinstanz $\langle w,1\rangle$ unklar ist, ob das Bedeutungsattribut von $\langle w,1\rangle$ abhängt oder nicht, wird die Berechnung ausgeführt.

■

Zulässige Dekoration und zulässige Termdekoration sind wohl zu unterscheiden. Während bei der zulässigen Dekoration jeder Attributinstanz eine *Funktion über Wertmengen* aus dem semantischen Bereich einer Makroattributgrammatik zugeordnet wird, wird bei der zulässigen Termdekoration ausschließlich mit *Termen über den Funktionssymbolen und Parametern* operiert, die auf den rechten Seiten der semantischen Gleichungen eines Ableitungsbaums vorkommen.

Jedoch existiert eine Abbildung $\hat{\varphi}$, die die zulässige Termdekoration $wert_t$ jedes Ableitungsbaums t auf eine zulässige Dekoration val_t von t abbildet, weshalb die Verwendbarkeit von $wert_t$ zur Repräsentation von val_t korrekt ist. Die Abbildung $\hat{\varphi}$ wird als homomorphe Fortsetzung der Interpretation φ auf Termen über Funktionssymbolen $f \in \Phi$ und Parametern $y \in Y$ definiert, wobei φ, Φ und Y aus der Spezifikation der Makroattributgrammatik stammen:

Definition 3.24 (Terminterpretation)
Für jeden Term $z \in T_\Phi(Y)^{\kappa_0}$ mit $var(z) = \{y_1^{\kappa_1}, \ldots, y_l^{\kappa_m}\}$ mit $m \geq 0$ und $\kappa_0, \ldots, \kappa_m \in \mathrm{K}$ und für alle $c_1 \in \Omega^{\kappa_1}, \ldots, c_m \in \Omega^{\kappa_m}$ ist

$$\begin{aligned}\hat{\varphi}(z, c_1, \ldots, c_m) =& c_i, \text{ falls } z = y_i \text{ und } y_i \in Y \\ =& \varphi(f)(\hat{\varphi}(z_1, c_1, \ldots, c_m), \ldots, \hat{\varphi}(z_k, c_1, \ldots, c_m)), \\ & \text{falls } z = f(z_1, \ldots, z_k) \text{ und } f \in \Phi^{(\kappa_1 \ldots \kappa_k, \kappa_0)} \\ & \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \\ & \text{ist } \kappa_i \in \mathrm{K} \text{ und } z_i \in T_\Phi(Y)^{\kappa_i}\end{aligned}$$

□

Das Lemma 3.25 setzt die Terminterpretation[5] und die Termsubstitution[6] in Beziehung. Es zeigt, daß die Reihenfolge der Anwendung von Terminterpretation und Termsubstitution austauschbar ist, und gehört deshalb zu den Voraussetzungen des Beweises zu Lemma 3.26.

Lemma 3.25
Für jedes $l \geq 0$, für alle $i \in \{1, \ldots, l\}$, für alle $\kappa_0, \kappa_i \in \mathrm{K}$, für alle $t \in T_\Phi(Y)^{\kappa_0}$ mit $var(t) = \{y_1^{\kappa_1}, \ldots, y_l^{\kappa_l}\}$ und für alle $t_i \in T_\Phi(Y)^{\kappa_i}$ und für jedes $m \geq 0$, für alle $j \in \{1, \ldots, m\}$, für alle $\iota_0, \iota_j \in \mathrm{K}$, für alle $c_j \in \Omega^{\iota_j}$ gilt:

$$\begin{aligned}&\hat{\varphi}(t[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m) \\ =&\hat{\varphi}(t, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m)).\end{aligned}$$

□

Beweis
Der Beweis erfolgt durch strukturelle Induktion über den Termaufbau von t.

Induktionsanfang: Sei $t = y_i$ mit $y_i \in Y$.

$$\begin{aligned}&\hat{\varphi}(t[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m) \\ =&\hat{\varphi}(y_i[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m)\end{aligned}$$

Nach Definition 2.8 gilt:

$$=\hat{\varphi}(\Xi(t_i), c_1, \ldots, c_m)$$

Nach Definition 3.24 gilt:

$$\begin{aligned}=&\hat{\varphi}(y_i, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m)) \\ =&\hat{\varphi}(t, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m))\end{aligned}$$

[5] siehe Definition 3.24
[6] siehe Definition 2.8

Induktionsschritt: Sei $t = f(s_1, \ldots, s_k)$ mit $f \in \Phi^{(\varsigma_1 \ldots \varsigma_k, \varsigma_0)}$ mit $k \geq 0$ und für alle $n \in \{1, \ldots, k\}$ ist $\varsigma_0, \varsigma_n \in \mathrm{K}$ und $s_n \in T_\Phi(Y)^{\varsigma_n}$.

$$\begin{aligned}
&\hat{\varphi}(t[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m) \\
=&\hat{\varphi}(f(s_1, \ldots, s_k)[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m)
\end{aligned}$$

Nach Definition 2.8 gilt:

$$\begin{aligned}
=&\hat{\varphi}(f(s_1[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], \ldots, \\
&\qquad s_k[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)]), c_1, \ldots, c_m)
\end{aligned}$$

Nach Definition 3.24 gilt:

$$\begin{aligned}
=&\varphi(f)(\hat{\varphi}(s_1[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m), \ldots, \\
&\qquad \hat{\varphi}(s_k[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m))
\end{aligned}$$

Nach Anwendung der *Induktionsvoraussetzung* auf jedes $\hat{\varphi}(s_n[y_1 := \Xi(t_1), \ldots, y_l := \Xi(t_l)], c_1, \ldots, c_m)$ gilt:

$$\begin{aligned}
=&\varphi(f)(\hat{\varphi}(s_1, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m)), \ldots, \\
&\qquad \hat{\varphi}(s_k, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m)))
\end{aligned}$$

Nach Definition 3.24 gilt:

$$\begin{aligned}
=&\hat{\varphi}(f(s_1, \ldots, s_k), \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m)) \\
=&\hat{\varphi}(t, \hat{\varphi}(\Xi(t_1), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi(t_l), c_1, \ldots, c_m))
\end{aligned}$$

Also gilt Lemma 3.25.

■

Lemma 3.26
Sei $M = (G_0, D, B, R)$ mit $G_0 = (N, \Sigma, Z, P)$ und $M \in NZMAG$. Sei $t \in DT(G_0)$, sei $x \in node(t)$ und $prod_t(x) = X_0 \to w_0 X_1 w_1 \ldots X_n w_n$ wobei $n \geq 0$. Für jedes $m \geq 0$, für alle $i \in \{1, \ldots, m\}$, für alle $\kappa_0, \kappa_i \in \mathrm{K}$, für alle $l \in \{0, \ldots, n\}$ und für alle $\langle a, x.l\rangle^{(\kappa_1 \ldots \kappa_m, \kappa_0)} \in A(t)$ gilt: Für alle $c_i \in \Omega^{\kappa_i}$ ist

$$val_t(\langle a, x.l\rangle)(c_1, \ldots, c_m) = \hat{\varphi}(wert_t(\langle a, x.l\rangle), c_1, \ldots, c_m)$$

wobei val_t zulässige Dekoration und $wert_t$ zulässige Termdekoration sind.

□

Beweis
Sei $\langle a, x.l\rangle \in A(t)$ mit $\langle a, x.l\rangle(y_1, \ldots, y_m) = [\zeta]_x \in R(t)$, wobei $[\zeta]_x \in T_{\Phi \cup A(t)}(Y)$ ist. Nach Definition 3.12 gilt:

$$val_t(\langle a, x.l\rangle)(c_1, \ldots, c_m) = \Theta([\zeta]_x, c_1, \ldots, c_m).$$

Nach Definition 3.22 gilt:

$$\hat{\varphi}(wert_t(\langle a, x.l\rangle), c_1, \ldots, c_m) = \hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m).$$

Aus der Gültigkeit von

$$\Theta([\zeta]_x, c_1, \ldots, c_m) = \hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m) \tag{3.2}$$

folgt Lemma 3.26.

Der Beweis der Gleichung 3.2 erfolgt durch Induktion über die topologische Sortierung des Abhängigkeitsgraphen.

Sei $D(t) = (A(t), E(t))$. Aus $M \in NZMAG$ folgt, daß eine topologische Sortierung von $D(t)$ existiert. Sei $topsort(D(t))$ eine topologische Sortierung von $D(t)$, und gelte für alle $\langle a', x'\rangle \in sym([\zeta]_x) \cap A(t)$, daß $pos_{topsort(D(t))}(\langle a', x'\rangle) < pos_{topsort(D(t))}(\langle a, x.l\rangle)$ ist.

Induktionsanfang: Sei $pos_{topsort(D(t))}(\langle a, x.l\rangle) = 1$. Dann muß $[\zeta]_x \in T_{\Phi\cup\emptyset}(\{y_1, \ldots, y_m\})$ gelten, da es kein $\langle a', x'\rangle \in A(t)$ mit $(\langle a', x'\rangle, \langle a, x.l\rangle) \in E(t)$ gibt.

Der Beweis der Gleichung 3.2 erfolgt durch strukturelle Induktion über den Termaufbau von $[\zeta]_x$.

Fall 1: Sei $[\zeta]_x = y_i$ und $y_i \in \{y_1, \ldots, y_m\}$.

$$\begin{aligned} &\Theta([\zeta]_x, c_1, \ldots, c_m) \\ =&\Theta(y_i, c_1, \ldots, c_m) \end{aligned}$$

Nach Definition 3.12 gilt:

$$=c_i$$

Nach Definition 3.24 gilt:

$$=\hat{\varphi}(y_i, c_1, \ldots, c_m)$$

Nach Definition 3.22 gilt:

$$\begin{aligned} =&\hat{\varphi}(\Xi(y_i), c_1, \ldots, c_m) \\ =&\hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m). \end{aligned}$$

Fall 2: Sei $[\zeta]_x = f([\zeta_1]_x, \ldots, [\zeta_r]_x)$ mit $r \geq 0$, $f \in \Phi^{(\kappa_1 \ldots \kappa_r, \kappa_0)}$, $\kappa_0, \ldots, \kappa_r \in \mathrm{K}$ und $[\zeta_1]_x^{\kappa_1}, \ldots, [\zeta_r]_x^{\kappa_r} \in T_{\Phi\cup A(t)}(Y)$.

$$\begin{aligned} &\Theta([\zeta]_x, c_1, \ldots, c_m) \\ =&\Theta(f([\zeta_1]_x, \ldots, [\zeta_r]_x), c_1, \ldots, c_m) \end{aligned}$$

Nach Definition 3.12 gilt:

$$=\varphi(f)(\Theta([\zeta_1]_x, c_1, \ldots, c_m), \ldots, \Theta([\zeta_r]_x, c_1, \ldots, c_m))$$

Für alle $[\zeta_1]_x, \ldots, [\zeta_r]_x \in T_{\Phi \cup \{\langle a', x'\rangle | pos_{topsort(D(t))}(\langle a', x'\rangle) \leq j\}}(\{y_1, \ldots, y_m\})$ gilt nach *Voraussetzung* der Induktion über die topologische Sortierung Gleichung 3.2. Daraus folgt:

$$=\varphi(f)(\hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Nach Definition 3.24 gilt:

$$=\hat{\varphi}(f(\Xi([\zeta_1]_x), \ldots, \Xi([\zeta_r]_x)), c_1, \ldots, c_m)$$

Nach Definition 3.22 gilt:

$$\begin{aligned}&=\hat{\varphi}(\Xi(f([\zeta_1]_x, \ldots, [\zeta_r]_x)), c_1, \ldots, c_m)\\&=\hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m).\end{aligned}$$

Induktionsschritt: Sei $pos_{topsort(D(t))}(\langle a, x.l\rangle) = j+1$ mit $j \in \{1, \ldots, |A(t)| - 1\}$. Dann ist $[\zeta]_x \in T_{\Phi \cup \{\langle a', x'\rangle | pos_{topsort(D(t))}(\langle a', x'\rangle) \leq j\}}(\{y_1, \ldots, y_m\})$.

Der Beweis der Gleichung 3.2 erfolgt durch strukturelle Induktion über den Termaufbau von $[\zeta]_x$.

Fall 1: Sei $[\zeta]_x = y_i$ und $y_i \in \{y_1, \ldots, y_m\}$.

$$\begin{aligned}&\Theta([\zeta]_x, c_1, \ldots, c_m)\\=&\Theta(y_i, c_1, \ldots, c_m)\end{aligned}$$

Nach Definition 3.12 gilt:

$$=c_i.$$

Nach Definition 3.24 gilt:

$$=\hat{\varphi}(y_i, c_1, \ldots, c_m)$$

Nach Definition 3.22 gilt:

$$\begin{aligned}&=\hat{\varphi}(\Xi(y_i), c_1, \ldots, c_m)\\&=\hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m).\end{aligned}$$

Fall 2: Sei $[\zeta]_x = f([\zeta_1]_x, \ldots, [\zeta_r]_x)$ mit $r \geq 0$, $f \in \Phi^{(\kappa_1 \ldots \kappa_r, \kappa_0)}$, $\kappa_0, \ldots, \kappa_r \in$ K und $[\zeta_1]_x^{\kappa_1}, \ldots, [\zeta_r]_x^{\kappa_r} \in T_{\Phi \cup A(t)}(Y)$.

$$\begin{aligned}&\Theta([\zeta]_x, c_1, \ldots, c_m)\\=&\Theta(f([\zeta_1]_x, \ldots, [\zeta_r]_x), c_1, \ldots, c_m)\end{aligned}$$

Nach Definition 3.12 gilt:

$$=\varphi(f)(\Theta([\zeta_1]_x, c_1, \ldots, c_m), \ldots, \Theta([\zeta_r]_x, c_1, \ldots, c_m)).$$

Für alle $[\zeta_1]_x, \ldots, [\zeta_r]_x \in T_{\Phi \cup \{\langle a', x' \rangle | pos_{topsort(D(t))}(\langle a', x' \rangle) \leq j\}}(\{y_1, \ldots, y_m\})$ gilt nach *Voraussetzung* der Induktion über die topologische Sortierung Gleichung 3.2. Daraus folgt:

$$=\varphi(f)(\hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Nach Definition 3.24 gilt:

$$=\hat{\varphi}(f(\Xi([\zeta_1]_x), \ldots, \Xi([\zeta_r]_x)), c_1, \ldots, c_m)$$

Nach Definition 3.22 gilt:

$$=\hat{\varphi}(\Xi(f([\zeta_1]_x, \ldots, [\zeta_r]_x)), c_1, \ldots, c_m)$$
$$=\hat{\varphi}(\Xi([\zeta]_x), c_1 \ldots, c_m).$$

Fall 3: Sei $[\zeta]_x = \langle a', x' \rangle([\zeta_1]_x, \ldots, [\zeta_r]_x)$ mit $r \geq 0$, $\langle a', x' \rangle^{(\kappa_1 \ldots \kappa_r, \kappa_0)} \in A(t)$, $\kappa_0, \ldots, \kappa_r \in \mathrm{K}$ und $[\zeta_1]_x^{\kappa_1}, \ldots, [\zeta_r]_x^{\kappa_r} \in T_{\Phi \cup A(t)}(Y)$.

$$\Theta([\zeta]_x, c_1, \ldots, c_m)$$
$$=\Theta(\langle a', x' \rangle([\zeta_1]_x, \ldots, [\zeta_r]_x), c_1, \ldots, c_m)$$

Aus Definition 3.12 folgt:

$$=val_t(\langle a', x' \rangle)(\Theta([\zeta_1]_x, c_1, \ldots, c_m), \ldots, \Theta([\zeta_r]_x, c_1, \ldots, c_m)).$$

Für alle $[\zeta_1]_x, \ldots, [\zeta_r]_x \in T_{\Phi \cup \{\langle a', x' \rangle | pos_{topsort(D(t))}(\langle a', x' \rangle) \leq j\}}(\{y_1, \ldots, y_m\})$ gilt nach *Voraussetzung* der Induktion über die topologische Sortierung Gleichung 3.2. Daraus folgt:

$$=val_t(\langle a', x' \rangle)(\hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Sei $\langle a', x' \rangle(y_1', \ldots y_{m'}') = [\zeta']_{x''} \in R(t)$.

$$=\Theta([\zeta']_{x''}, \hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Wegen $[\zeta']_{x''} \in T_{\Phi \cup \{\langle a', x' \rangle | pos_{topsort(D(t))}(\langle a', x' \rangle) \leq j\}}(\{y_1, \ldots, y_m\})$ gilt nach *Voraussetzung* der Induktion über die topologische Sortierung Gleichung 3.2. Daraus folgt:

$$=\hat{\varphi}(\Xi([\zeta']_{x''}), \hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Aus Definition 3.22 folgt:

$$=\hat{\varphi}(wert_t(\langle a', x' \rangle), \hat{\varphi}(\Xi([\zeta_1]_x), c_1, \ldots, c_m), \ldots, \hat{\varphi}(\Xi([\zeta_r]_x), c_1, \ldots, c_m))$$

Nach Lemma 3.25 gilt:

$$=\hat{\varphi}(wert_t(\langle a', x' \rangle)[y_1 := \Xi([\zeta_1]_x), \ldots, y_r := \Xi([\zeta_r]_x)], c_1 \ldots, c_m)$$

Aus Definition 3.22 folgt:

$$\begin{aligned}&=\hat{\varphi}(\Xi(\langle a', x'\rangle([\zeta_1]_x, \dots, [\zeta_r]_x)), c_1 \dots, c_m)\\&=\hat{\varphi}(\Xi([\zeta]_x), c_1 \dots, c_m).\end{aligned}$$

Daraus folgt, daß Gleichung 3.2 für alle Attributinstanzen gilt. Also gilt Lemma 3.26.

■

Speziell für die Instanz des Bedeutungsattributs α_0 eines Ableitungsbaums t folgt aus Lemma 3.26 die Gültigkeit von

$$val_t(\langle \alpha_0, \varepsilon\rangle) = \hat{\varphi}(wert_t(\langle \alpha_0, \varepsilon\rangle)),$$

woraus für jede Makroattributgrammatik MAG folgt

$$\tau_{tv}(MAG) = \{(t, \hat{\varphi}(wert_t(\langle \alpha_0, \varepsilon\rangle))) \mid t \in DT(G_0), wert_t \text{ ist zulässige Termdekoration } \}.$$

Beispiel 3.27 (Fortsetzung von 3.23)
Die Anwendung von $\hat{\varphi}$ auf $wert_{t_1}$ liefert

$$\begin{aligned}&\hat{\varphi}(wert_{t_1}(\langle \alpha_0, \varepsilon\rangle))\\&\qquad = \hat{\varphi}(s(null))\\&\qquad = \varphi(s)(\hat{\varphi}(null))\\&\qquad = \varphi(s)(\varphi(null)())\\&\qquad = \varphi(s)(0)\\&\qquad = 1\end{aligned}$$

und damit ist $(t_1, 1) \in \tau_{tv}(MAG_{10})$.

■

Orientiert an den Zielen der vorliegenden Arbeit, die die geordneten Makroattributgrammatiken diskutieren möchte, wird an dieser Stelle *kein* konkreter Auswerter für nichtzirkuläre Makroattributgrammatiken dargestellt. Dagegen wird im Kapitel 4 ein speziell auf die Eigenschaften der geordneten Makroattributgrammatiken zugeschnittener Auswerter konstruiert.

Kapitel 4

Geordnete Makroattributgrammatiken

Intuitiv enthält die Klasse der geordneten Makroattributgrammatiken diejenigen Makroattributgrammatiken, für deren Nichtterminalsymbole eine Ordnung ihrer Attribute aus der Spezifikation der Makroattributgrammatik abgeleitet werden kann. Sinn und Zweck der Ordnung der Attribute ist die Konstruktion von *optimierten* traversierenden Auswertern.

Die Existenz solcher Ordnungen ist abhängig von der Struktur der Abhängigkeitsgraphen der Produktionen, weshalb wegen Beobachtung 3.2 geordnete Makroattributgrammatiken als Erweiterung geordneter Attributgrammatiken aufgefaßt werden, die von Kastens in [Kas80] eingeführt wurden.

Existiert für jedes Nichtterminalsymbol X eine Ordnung der Attribute von X, so kann die Entscheidungsfindung eines traversierenden Auswerters über den nächsten zu gehenden Schritt optimiert werden: Eine Ordnung der Attribute von X induziert eine Ordnung auf den Attributinstanzen *jedes* Knotens eines Ableitungsbaums, der mit X beschriftet ist. Dadurch legt die Ordnung der Attribute von X eindeutig eine Reihenfolge fest, in der die Attributinstanzen jedes mit X beschrifteten Knotens garantiert ausgewählt und ausgewertet werden können, weshalb sie als *Auswertungsordnung von X* bezeichnet wird. Darüberhinaus ist die Auswertungsordnung eines Nichtterminalsymbols universell, das heißt für alle Ableitungsbäume gleichermaßen gültig.

Die Auswertungsordnung des Nichtterminalsymbols, das in einer Produktion vorkommt, induziert eine Ordnung auf den Attributvorkommen der Produktion mit identischen Indizes. Der Abhängigkeitsgraph der Produktion induziert eine weitere Ordnung auf den Attributvorkommen der Produktion. Aus dem gleichzeitigen Vorhandensein beider Ordnungen läßt sich eine Ordnung über allen Attributvorkommen der Produktion ableiten, die sowohl den Auswertungsordnungen der Nichtterminalsymbole als auch dem Abhängigkeitsgraphen genügt. Diese abgeleitete *Auswertungsordnung der Produktion* legt eine Reihenfolge fest, in der die Attributinstanzen der Knoten, die durch Anwendung der Produktion entstehen, garantiert ausgewertet werden können. Darüberhinaus ist in der Auswertungsordnung einer Produktion die notwendige Reihenfolge der Knotenwechsel zwischen dem Knoten, an dem die Produktion angewendet wurde, und seinem Vorgängerknoten und seinen Nachfolgerknoten implizit enthalten. Bei geeigneter Kodierung beschreibt die Auswertungsordnung einer Produktion p den „Arbeitsplan“, den ein traversierender Auswerter abarbeiten muß, wenn er einen Knoten erreicht, an dem die Produktion p angewendet wurde.

4.1 Abhängigkeitsgraphen und Auswertungsordnungen

Die Anpassung der Konzepte aus dem Bereich der geordneten Attributgrammatiken basiert auf der Darstellung von Attributgrammatiken in [Vog95] und resultiert in den Definitionen 4.1, 4.2 und 4.4.

Definition 4.1 (Oben und unten erweiterter Abhängigkeitsgraph)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $p : X_0 \to w_0 X_1 w_1 \dots X_n w_n$, $n \geq 0$, eine Produktion von G_0. Sei für alle $i \in \{0, \dots, n\}$ ein Graph $D_i = (A(X_i), \to_i)$ mit $\to_i \subseteq A(X_i) \times A(X_i)$ über den Attributen von X gegeben. Sei $D(p) = (A(p), E(p))$ der Abhängigkeitsgraph von p. Der *oben und unten erweiterte Abhängigkeitsgraph von* p, bezeichnet mit $D(p)[D_0; D_1, \dots, D_n]$, ist der Graph $(A(p), E)$ mit der Kantenrelation

$$E = E(p) \cup \{(\langle a, i\rangle, \langle b, i\rangle) \mid (a, b) \in \to_i, i \in \{0, \dots, n\}\}$$

□

Für die Diskussion geordneter Makroattributgrammatiken interessieren ausschließlich jene Graphen über den Attributen der Nichtterminalsymbole, die ihrerseits aus den Abhängigkeitsgraphen der Produktionen abgeleitet werden können.

Definition 4.2 (is/si-Graphen der Nichtterminalsymbole)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $N = \{X_1, \dots, X_m\}$ die Menge der Nichtterminalsymbole von G_0. Die *is/si-Graphen von* $X_1, \dots, X_m$, bezeichnet mit $is/si(X_1), \dots, is/si(X_m)$, sind die kleinsten Graphen

$$g_1 = (A(X_1), E(X_1)), \dots, g_m = (A(X_m), E(X_m)),$$

für die gilt: Für jede Produktion $p : X_{i_0} \to w_0 X_{i_1} w_i \dots X_{i_n} w_n$, $i_0, \dots, i_n \in \{1, \dots, m\}$ und für alle $j \in \{0, \dots, n\}$ gilt:
Wenn $(\langle a, j\rangle, \langle b, j\rangle) \in E^+$, wobei E die Kantenrelation von $D(p)[g_{i_0}; g_{i_1}, \dots, g_{i_n}]$ ist, dann ist $(a, b) \in E(X_{i_j})$.

□

Abbildung 4.1 zeigt die is/si-Graphen der Nichtterminalsymbole Z und L der Beispielgrammatik MAG_{10}. Die um die is/si-Graphen nach oben und unten erweiterten Abhängigkeitsgraphen sind in Abbildung 4.2 dargestellt.

Ein Algorithmus zur Berechnung der is/si-Graphen einer gegebenen Makroattributgrammatik ist der Algorithmus 4.3, der in Analogie zu [Vog95] aufgestellt wurde.

Algorithmus 4.3 (Berechnung der is/si-Graphen)
Eingabe : Makroattributgrammatik $MAG = (G_0, D, B, R)$, $G_0 = (N, \Sigma, Z, P)$;

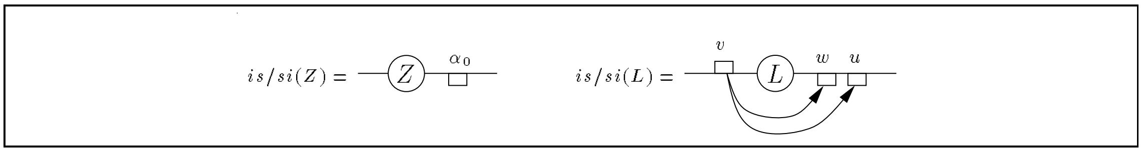


Abbildung 4.1: Die Graphen $is/si(Z)$ und $is/si(L)$

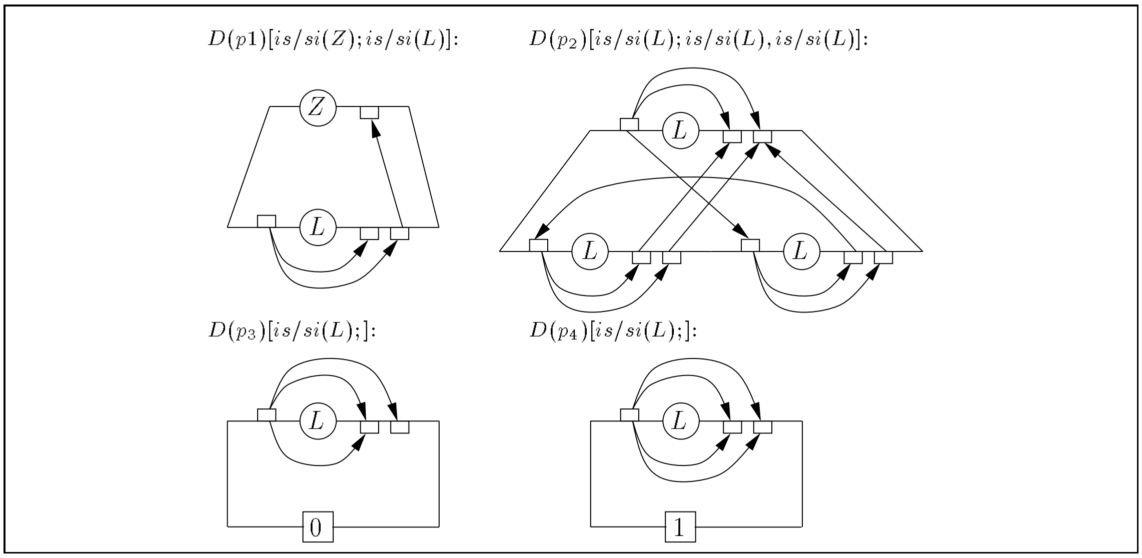


Abbildung 4.2: Nach oben und unten erweiterte Abhängigkeitsgraphen

Ausgabe : für jedes $X \in N$ den Graphen $is/si(X)$;
Variable : $\$[X] = (\$A(X), \$E(X))$ zur Aufnahme von $is/si(X)$;

foreach $X \in N$ **do** $\$[X] := (A(X), \emptyset)$ **end** ;
repeat
  foreach $p : X_0 \to w_0 X_1 w_1 \ldots X_n w_n \in P$ **do**
    for $j := 0$ **to** n **do**
      if es gibt $(\langle a, j\rangle, \langle b, j\rangle) \in E^+$
        wobei E Kantenrelation von $D(p)[\$[X_0]; \$[X_1], \ldots, \$[X_n]]$ ist
      then
        $\$[X_j] := (A(X_j), \$E(X_j) \cup \{(a, b)\})$
      end
    end
  end
until es gibt keine Änderung eines Graphen $\$[X]$
  gegenüber dem vorangegangenen Durchlauf;
foreach $X \in N$ **do** $is/si(X) := \$[X]$ **end**

□

Der is/si-Graph eines Nichtterminalsymbols X induziert eine Reihenfolge, in der die Werte der Attributinstanzen eines Knotens, der mit X beschriftet ist, berechnet werden können. Um diese Reihenfolge zu bestimmen, muß die Halbordnung, die die Kantenrelation von $is/si(X)$ darstellt, in eine totale Ordnung transformiert werden, die der Kantenrelation genügt. Die resultierende Ordnungsrelation wird als *Auswertungsordnung von X* bezeichnet.

Tatsächlich liefert jede topologische Sortierung von $is/si(X)$ eine totale Ordnung über den Attributen in $A(X)$. Unter allen topologischen Sortierungen sind jedoch diejenigen von besonderem Interesse, in der diejenigen Attribute unmittelbar aufeinanderfolgen, die von derselben Teilmenge von $A(X)$ *unabhängig* sind. Durch diese spezielle Sortierung entsteht eine Ordnung mit alternierenden Abschnitten aufeinanderfolgender inheriter Attribute und aufeinanderfolgender synthetischer Attribute, wobei die Anzahl der Abschnitte und damit die Anzahl der Wechsel zwischen Abschnitten minimiert ist. Unter dem Blickwinkel der beabsichtigten Konstruktion eines traversierenden Auswerters wird der Sinn dieser Minimierung deutlich: Die Werte der Instanzen inheriter Attribute an einem Knoten x werden durch semantische Gleichungen „oberhalb“ von x bestimmt. Diese semantischen Gleichungen resultieren aus Instanziierungen der semantischen Regeln der Produktion, die am Vorgängerknoten $x.$-1 von x angewendet wurde. Demgegenüber werden die Werte der Instanzen synthetischer Attribute am Knoten x durch Gleichungen bestimmt, die aus Regeln derjenigen Produktion abgeleitet werden, die an x selbst angewendet wurde. Um die unterschiedlichen Kontexte für die Berechnung der Werte der Instanzen inheriter Attribute beziehungsweise synthetischer Attribute am Knoten x herzustellen, muß der Auswerter zum Knoten $x.$-1 beziehungsweise zum Knoten x wechseln. Die Anzahl der notwendigen Wechsel zwischen einem Knoten x, der mit X beschriftet ist, und seinem Vorgänger $x.$-1 ist demzufolge durch die Anzahl der Wechsel zwischen den Abschnitten der Auswertungsordnung von X bestimmt.

Definition 4.4 (Auswertungsordnung eines Nichtterminalsymbols)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $X \in N$ ein Nichtterminalsymbol von G_0. Sei $is/si(X) = (A(X), E(X))$ der is/si-Graph von X. Eine *aus $is/si(X)$ abgeleitete Auswertungsordnung von X*, bezeichnet mit T_X, ist eine totale Ordnung

$$\tilde{\beta}_1 \tilde{\alpha}_1 \ldots \tilde{\beta}_{k(X)} \tilde{\alpha}_{k(X)}$$

auf $A(X)$ mit minimalem $k(X) \in I\!N$, sodaß für jedes $j \in \{1, \ldots, k(X)\}$ gilt:

- $\tilde{\beta}_j$ ist eine beliebige totale Anordnung aller inheriten Attribute, die Maximalelemente der Relation

  $$E(X)\lceil A(X) \setminus \{a \mid a \in A(X), a \text{ tritt in } \tilde{\alpha}_j \tilde{\beta}_{j+1} \tilde{\alpha}_{j+1} \ldots \tilde{\beta}_{k(X)} \tilde{\alpha}_{k(X)} \text{ auf } \}\rceil$$

  sind, und

- $\tilde{\alpha}_j$ ist eine beliebige totale Anordnung aller synthetischen Attribute, die Maximalelemente der Relation

  $$E(X)\lceil A(X) \setminus \{a \mid a \in A(X), a \text{ tritt in } \tilde{\beta}_{j+1} \tilde{\alpha}_{j+1} \ldots \tilde{\beta}_{k(X)} \tilde{\alpha}_{k(X)} \text{ auf } \}\rceil$$

  sind.

□

Beispiel 4.5 (Fortsetzung von 3.5)
Die Auswertungsordnungen T_Z und T_L der Nichtterminalsymbole Z und L seien:

$$\begin{array}{ll} T_Z = \tilde{\beta}_Z \tilde{\alpha}_Z & \text{mit } \tilde{\beta}_Z = \varepsilon, \quad \tilde{\alpha}_Z = \alpha_0 \\ T_L = \tilde{\beta}_L \tilde{\alpha}_L & \text{mit } \tilde{\beta}_L = v, \quad \tilde{\alpha}_L = wu \end{array}$$

(Wegen $k(Z) = 1$ und $k(L) = 1$ werden zur Vereinfachung der Unterscheidung der totalen Anordnungen die Nichtterminalsymbole zur Indizierung verwendet.) ■

Beobachtung 4.6
Die Bestimmung einer Auswertungsordnung ist ebenfalls über die totalen Anordnungen der *Minimalelemente* von eingeschränkten Kantenrelationen möglich.

Intuitiv beschreiben Ordnungen, die aus Anordnungen von Minimalelementen bestehen, eine Auswertungsreihenfolge der Attribute, in der jedes Attribut *so früh wie möglich* berücksichtigt wird. Hingegen wird in Definition 4.4 eine Auswertungsreihenfolge spezifiziert, in der jedes Attribut *so spät wie möglich* eingeordnet wird.

□

Eine Auswertungsordnung T_X eines Nichtterminalsymbols $X \in N$ muß nicht existieren. Insbesondere dann, wenn der Graph $is/si(X)$ zyklisch ist, kann keine Auswertungsordnung aus $is/si(X)$ abgeleitet werden.

Die algorithmische Umsetzung der Definition 4.4 erfolgt mit Algorithmus 4.7.[1] Das auffälligste Merkmal des Algorithmus 4.7 ist die umgekehrte Reihenfolge, in der die totalen Anordnungen von inheriten und synthetischen Attributen berechnet werden, die die Auswertungsordnung eines Nichtterminalsymbols bilden. Diese Berechnungsreihenfolge resultiert aus der effizienten Suche nach den Maximalelementen der Einschränkungen der Kantenrelation: Im ersten Durchlauf lassen sich am leichtesten die Maximalelemente der uneingeschränkten Kantenrelation bestimmen; das sind alle Attribute, von denen keine Kante ausgeht. Die gefundenen Maximalelemente werden von der Attributmenge abgezogen und die Kantenrelation auf die verbleibenden Attribute eingeschränkt. Die Berechnung wird mit der eingeschränkten Kantenrelation wiederholt, solange bis alle Attribute betrachtet wurden.

Um die korrekte Reihenfolge der totalen Anordnungen auszugeben, werden vor Rückkehr des Algorithmus die berechneten Anordnungen nach fallendem Index angeordnet.

Algorithmus 4.7 (Ableitung der Auswertungsordnungen)
Eingabe : eine Makroattributgrammatik $MAG = (G_0, D, B, R), G_0 = (N, \Sigma, Z, P)$
für jedes $X \in N$ der Graph $is/si(X) = (A(X), E(X))$
Ausgabe : für jedes $X \in N$ ein $k(X) \in I\!N$ und
die Auswertungsordnung $T_X = \tilde{\beta}_1 \tilde{\alpha}_1 \ldots \tilde{\beta}_{k(X)} \tilde{\alpha}_{k(X)}$,
falls für jedes $X \in N$ ein T_X existiert
Variable : $\$A$, $\$E$ zur Aufnahme genau eines is/si-Graphen
k zur Aufnahme der Anzahl notwendiger Partitionen
$\$\beta_k$ zur Aufnahme der k-ten Anordnung inheriter Attribute und
$\$\alpha_k$ zur Aufnahme der k-ten Anordnung synthetischer Attribute

foreach $X \in N$ **do**
  $k := 0$; $\$A := A(X)$; $\$E := E(X)$;
  repeat
    $k := k + 1$; $\$\alpha_k := \varepsilon$; $\$\beta_k := \varepsilon$;
    foreach $a \in \$A \cap S{-}Att$ **do**
      if Es gibt kein $b \in \$A$ mit $(a, b) \in \$E$ **then** $\$\alpha_k := \$\alpha_k \circ a$; **end**
    end
    foreach a in $\$\alpha_k$ **do** $\$A := \$A \setminus \{a\}$; **end**
    $\$E := \$E\lceil\$A\rceil$;
    foreach $a \in \$A \cap I{-}Att$ **do**
      if Es gibt kein $b \in \$A$ mit $(a, b) \in \$E$ **then** $\$\beta_k := \$\beta_k \circ a$; **end**
    end
    foreach a in $\$\beta_k$ **do** $\$A := \$A \setminus \{a\}$; **end**
    $\$E := \$E\lceil\$A\rceil$;
  until $(\$A = \emptyset)$ **or** $(\$\beta_k = \varepsilon$ **and** $\$\alpha_k = \varepsilon)$
  if $\$A \neq \emptyset$ **then**
    (* Attribute in $\$A$ haben Nachfolger in $\$E$ *)
    write („Der Graph $is/si(X)$ ist zyklisch.“);

[1] Vergleiche den Algorithmus *Step3* in [RT89, Seite 260], der seinerseits eine Adaption des Algorithmus *Step3* ist, den Kastens in [Kas80, Seite 253] angibt.

```
            return;
        end
        (* Zuweisung mit Umkehrung der Reihenfolge der Anordnungen *)
        k(X) := k;
        T_X := $β_k$α_k ... $β_1$α_1;
    end
```

□

Der Algorithmus 4.7 leistet mehr als die Berechnung der totalen Anordnungen. Er testet zusätzlich die *is/si*-Graphen der Nichtterminalsymbole auf eventuell vorhandene Zyklen. Ist die Schleife zur Berechnung der Auswertungsordnung eines beliebigen X einmal durchlaufen worden, ohne daß mindestens ein Maximalelement in der eingeschränkten Kantenrelation des *is/si*-Graphen von X gefunden werden konnte, so ist der *is/si*-Graph zyklisch. In diesem Fall existiert keine Auswertungsordnung, der Algorithmus hält mit einer entsprechenden Meldung an.

Existieren die Auswertungsordnungen der Nichtterminalsymbole, so existiert für jede Produktion $p : X_0 \to w_0 X_1 w_1 \dots X_n w_n$ der nach oben und unten *um die Auswertungsordnungen* $T_{X_0}, \dots, T_{X_n}$ erweiterte Abhängigkeitsgraph der Produktion. Dieser Graph wird mit $D(p)[T_{X_0}; T_{X_1}, \dots, T_{X_n}]$ bezeichnet. Die Notation ist eine abkürzende Notation für die Erweiterung von $D(p)$ um die *Graphen*, die durch die Auswertungsordnungen induziert werden.

Beispiel 4.8 (Fortsetzung von 3.5)
Die um die Auswertungsordnungen T_Z und T_L der Nichtterminalsymbole Z und L nach oben und unten erweiterten Abhängigkeitsgraphen der Produktionen sind in Abbildung 4.3 dargestellt.

■

Die Struktur der um die Auswertungsordnungen erweiterten Abhängigkeitsgraphen der Produktionen bestimmt die Zugehörigkeit einer Makroattributgrammatik zur Klasse der geordneten Makroattributgrammatiken.

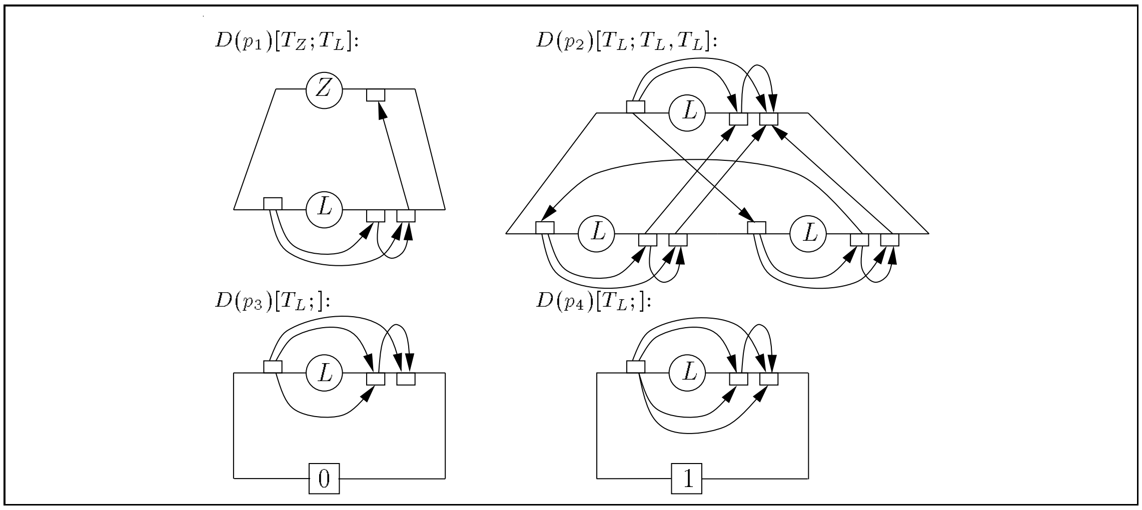


Abbildung 4.3: Die um die Auswertungsordnungen T_Z und T_L nach oben und unten erweiterten Abhängigkeitsgraphen

4.2 Definition und Einordnung

Die Definition der geordneten Makroattributgrammatiken ist eine Adaption der Definition der geordneten Attributgrammatiken aus [Vog95], da die in der vorliegenden Arbeit verwendete Terminologie auf [Vog95] zurückgeht. Ursprünglich wurden geordnete Attributgrammatiken jedoch von Kastens in [Kas80] eingeführt.

Definition 4.9 (Geordnete Makroattributgrammatik)
Eine Makroattributgrammatik $MAG = (G_0, D, B, R)$ heißt genau dann *geordnet*, wenn für jedes Nichtterminalsymbol X von G_0 eine aus $is/si(X)$ abgeleitete Auswertungsordnung T_X existiert, sodaß für alle Produktionen $p : X_0 \rightarrow w_0X_1w_1 \ldots X_nw_n$ von G_0 gilt, der Graph $D(p)[T_{X_0}; T_{X_1}, \ldots, T_{X_n}]$ ist ein azyklischer Graph.

Die Klasse der geordneten Makroattributgrammatiken wird mit $OMAG$ bezeichnet.

□

Lemma 4.10 (Hierarchie)
Die Menge der geordneten Makroattributgrammatiken ist eine echte Teilmenge der nichtzirkulären Makroattributgrammatiken: $OMAG \subset NZMAG$.

□

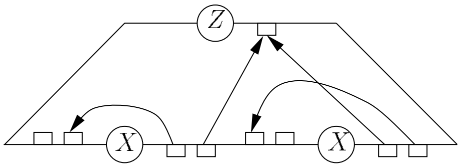
Beweis
Die Annahme, daß eine geordnete Makroattributgrammatik $M = (G_0, D, B, R)$ mit $G_0 = (N, \Sigma, Z, P)$ existiert, die nicht zur Klasse $NZMAG$ gehört, führt zu folgendem Widerspruch: Da M geordnet ist, existiert nach Definition 4.9 für jedes $X \in N$ eine Auswertungsordnung T_X. Eine Auswertungsordnung T_X existiert nach Definition 4.4 dann und nur dann, wenn der Graph $is/si(X)$ azyklisch ist.

Da M nicht zu $NZMAG$ gehört, gibt es nach Definition 3.16 einen Ableitungsbaum, dessen Abhängigkeitsgraph zyklisch ist. Dann muß es eine Produktion $p : X_0 \rightarrow w_0X_1w_1 \ldots X_nw_n$

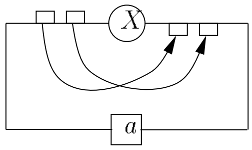
mit $n \geq 0$ geben, für die gilt, daß der nach oben und unten um die is/si-Graphen der Nichtterminalsymbole $X_0 \ldots X_n$ erweiterte Abhängigkeitsgraph von p zyklisch ist. Dann muß es ein Nichtterminalsymbol X_i, $i \in \{0, \ldots, n\}$, mit einem Graphen $is/si(X_i)$ geben, der zyklisch ist, und damit im Widerspruch zum Geordnetsein der Makroattributgrammatik steht.

Die Existenz einer nichtzirkulären Makroattributgrammatik, die nicht zur Klasse der geordneten Makroattributgrammatiken gehört, zeigt eine Beispielgrammatik mit folgenden Abhängigkeitsgraphen der Produktionen:

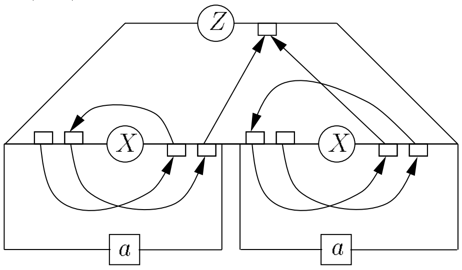
$D(p_Z)$:



$D(p_X)$:

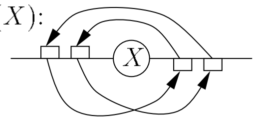


$D(t_{aa})$:



$D(t_{aa})$ ist azyklisch.

$is/si(X)$:



$is/si(X)$ ist zyklisch.

Demzufolge gilt Lemma 4.10. ∎

Beispiel 4.11 (Fortsetzung von 3.5)
Alle in Abbildung 4.3 dargestellten Graphen sind azyklisch, und andere lassen sich nicht für die Produktionen von MAG_{10} konstruieren, sodaß gilt: $MAG_{10} \in OMAG$.

∎

4.3 Auswerter für geordnete Makroattributgrammatiken

Wegen der Existenz der Auswertungsordnungen ist die Reihenfolge der Berechnung der Werte der Attributinstanzen zweier Knoten, die mit dem gleichen Nichtterminalsymbol beschriftet sind, nicht von den an den beiden Knoten angewendeten Produktionen, sondern nur von den Knotenbeschriftungen abhängig und damit identisch.

Traversierende Auswerter werden durch „Arbeitspläne“ gesteuert, die der Terminologie von [KW76] und [Vog95] folgend als *Pläne* bezeichnet werden. Ein Plan einer Produktion p läßt sich über mehrere Zwischenstufen aus dem nach oben und unten erweiterten Abhängigkeitsgraphen $D(p)[T_{X_0}; T_{X_1}, \ldots, T_{X_n}]$ ableiten.

Zunächst werden die gefundenen Auswertungsordnungen der Nichtterminalsymbole auf Anordnungen von Attributvorkommen abgebildet, wobei die Aufteilung der Auswertungsordnungen in totale Anordnungen $\tilde{\beta}_i$ und $\tilde{\alpha}_i$, $i \in \{1, \ldots, k(X)\}$, $X \in N$, für die Attributvorkommen übernommen wird.

Definition 4.12 (Anordnung von Attributvorkommen einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine geordnete Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $p : X_0 \to w_0 X_1 w_1 \ldots X_n w_n \in P$. Seien $T_{X_0}, \ldots, T_{X_n}$ die Auswertungsordnungen der Nichtterminalsymbole, die in p vorkommen.

Für jedes $j \in \{0, \ldots, n\}$, für jedes $i \in \{1, \ldots, k(X_j)\}$ und für jede totale Anordnung $\tilde{\beta}_i = b_1 \ldots b_l$ inheriter Attribute und für jede totale Anordnung $\tilde{\alpha}_i = a_1 \ldots a_m$ synthetischer Attribute, die in T_{X_j} auftreten, ist

$$\begin{aligned} \langle \tilde{\beta}_i, j \rangle &= \langle b_1, j \rangle \ldots \langle b_l, j \rangle, \text{ beziehungsweise} \\ \langle \tilde{\alpha}_i, j \rangle &= \langle a_1, j \rangle \ldots \langle a_m, j \rangle \end{aligned}$$

eine *Anordnung von Attributvorkommen von p.*

Die *Menge aller Anordnungen von Attributvorkommen von p* wird mit $\tilde{A}(p)$ bezeichnet.

□

Beispiel 4.13 (Fortsetzung von 3.5)
Für die Makroattributgrammatik MAG_{10} können folgende Mengen von Anordnungen von Attributvorkommen für die einzelnen Produktionen konstruiert werden:

$$\begin{aligned}
\tilde{A}(p_1) = \{\, \langle \tilde{\beta}_Z, 0 \rangle &= \varepsilon, \\
\langle \tilde{\alpha}_Z, 0 \rangle &= \langle \alpha_0, 0 \rangle, \\
\langle \tilde{\beta}_L, 1 \rangle &= \langle v, 1 \rangle, \\
\langle \tilde{\alpha}_L, 1 \rangle &= \langle w, 1 \rangle \langle u, 1 \rangle \} \\
\tilde{A}(p_2) = \{\, \langle \tilde{\beta}_L, 0 \rangle &= \langle v, 0 \rangle, \\
\langle \tilde{\alpha}_L, 0 \rangle &= \langle w, 0 \rangle \langle u, 0 \rangle, \\
\langle \tilde{\beta}_L, 1 \rangle &= \langle v, 1 \rangle, \\
\langle \tilde{\alpha}_L, 1 \rangle &= \langle w, 1 \rangle \langle u, 1 \rangle, \\
\langle \tilde{\beta}_L, 2 \rangle &= \langle v, 2 \rangle, \\
\langle \tilde{\alpha}_L, 2 \rangle &= \langle w, 2 \rangle \langle u, 2 \rangle \} \\
\tilde{A}(p_3) = \{\, \langle \tilde{\beta}_L, 0 \rangle &= \langle v, 0 \rangle, \\
\langle \tilde{\alpha}_L, 0 \rangle &= \langle w, 0 \rangle \langle u, 0 \rangle \} \\
\tilde{A}(p_4) = \{\, \langle \tilde{\beta}_L, 0 \rangle &= \langle v, 0 \rangle, \\
\langle \tilde{\alpha}_L, 0 \rangle &= \langle w, 0 \rangle \langle u, 0 \rangle \}
\end{aligned}$$

■

Intuitiv beschreiben die Anordnungen der Attributvorkommen „Boxen“ von Attributvorkommen, die jeweils diejenigen Attributvorkommen beinhalten, die von der gleichen Teilmenge der Attributvorkommen einer Produktion abhängig, jedoch untereinander unabhängig sind.

Die Abhängigkeiten zwischen den Anordnungen der Attributvorkommen einer Produktion stammen aus verschiedenen Quellen:

1. dem Abhängigkeitsgraph der Produktion,
2. der Reihenfolge der totalen Anordnungen von Attributvorkommen mit gleichem Index und
3. dem beabsichtigten Traversierungsschema: Die Traversierung jedes Teilbaums beginnt und endet stets beim Wurzelknoten des Teilbaums. Diese Traversierung induziert zusätzliche Abhängigkeiten ausgehend von der ersten Anordnung $\langle\tilde{\beta}_1, 0\rangle$ von Vorkommen inheriter Attribute des Nichtterminals auf der linken Seite einer Produktion zu jeder Anordnung $\langle\tilde{\beta}_1, j\rangle$, mit $j \in \{1, \dots, n\}$, und ausgehend von jeder Anordnung $\langle\tilde{\alpha}_{k(X_j)}, j\rangle$ zu $\langle\tilde{\alpha}_{k(X_0)}, 0\rangle$.

Die aufgezählten Abhängigkeiten zwischen den Anordnungen der Attributvorkommen einer Produktion werden durch den *Auswertungsgraphen der Produktion* beschrieben.

Definition 4.14 (Auswertungsgraph einer Produktion)
Sei $p : X_0 \to w_0X_1w_1 \dots X_nw_n \in P$. Seien $T_{X_0}, \dots, T_{X_n}$ die Auswertungsordnungen der Nichtterminalsymbole, die in p vorkommen. Sei $\tilde{A}(p)$ die Menge der Anordnungen der Attributvorkommen von p. Sei $D(p) = (A(p), E(p))$ der Abhängigkeitsgraph von p. Der *Auswertungsgraph von p*, bezeichnet mit $\tilde{D}(p)$, ist der Graph $\tilde{D}(p) = (\tilde{A}(p), \tilde{E}(p))$ über den Anordnungen von Attributvorkommen mit der Kantenrelation

$$\tilde{E}(p) = \tilde{E}_1(p) \cup \tilde{E}_2(p) \cup \tilde{E}_3(p)$$

wobei gilt:

$$\begin{aligned}
\tilde{E}_1(p) = \{ & (\langle\tilde{\gamma}_{i_1}, j_1\rangle, \langle\tilde{\gamma}_{i_2}, j_2\rangle) \mid j_1, j_2 \in \{0, \dots, n\}, 1 \le i_1 \le k(X_{j_1}), 1 \le i_2 \le k(X_{j_2}) \\
& \text{und es gibt } (\langle a, j_1\rangle, \langle b, j_2\rangle) \in E(p) \\
& \text{und } \langle a, j_1\rangle \text{ tritt in } \langle\tilde{\gamma}_{i_1}, j_1\rangle \text{ und } \langle b, j_2\rangle \text{ in } \langle\tilde{\gamma}_{i_2}, j_2\rangle \text{ auf } \} \\
\tilde{E}_2(p) = \{ & (\langle\tilde{\beta}_i, j\rangle, \langle\tilde{\alpha}_i, j\rangle) \mid j \in \{0, \dots, n\}, i \in \{1, \dots, k(X_j)\}\} \\
& \cup \{(\langle\tilde{\alpha}_i, j\rangle, \langle\tilde{\beta}_{i+1}, j\rangle) \mid j \in \{0, \dots, n\}, i \in \{1, \dots, k(X_j) - 1\}\} \\
\tilde{E}_3(p) = \{ & (\langle\tilde{\beta}_1, 0\rangle, \langle\tilde{\beta}_1, j\rangle) \mid j \in \{1, \dots, n\}\} \\
& \cup \{(\langle\tilde{\alpha}_{k(X_j)}, j\rangle, \langle\tilde{\alpha}_{k(X_0)}, 0\rangle) \mid j \in \{1, \dots, n\}\}
\end{aligned}$$

($\tilde{E}_1(p)$ enthält die Kanten, die durch den Abhängigkeitsgraphen induziert werden. $\tilde{E}_2(p)$ enthält die Kanten, die durch die Auswertungsordnungen induziert werden. $\tilde{E}_3(p)$ enthält die Kanten, die durch die Traversierungsreihenfolge induziert werden.) Die Kantenrelation $\tilde{E}(p)$ läßt sich in zwei disjunkte Teilmengen zerlegen:

- in die Menge der *Innenkanten des Auswertungsgraphen*:

$$\{(\langle\tilde{\gamma}, i\rangle, \langle\tilde{\delta}, j\rangle) \mid \langle\tilde{\delta}, j\rangle \text{ ist Anordnung von Innenattributvorkommen } \}$$

  und

- in die Menge der *Außenkanten des Auswertungsgraphen*:

$$\{(\langle\tilde{\gamma}, i\rangle, \langle\tilde{\delta}, j\rangle) \mid \langle\tilde{\delta}, j\rangle \text{ ist Anordnung von Außenattributvorkommen } \}$$

□

Intuitiv beschreiben die Innenkanten des Auswertungsgraphen einer Produktion p solche Abhängigkeiten zwischen Attributinstanzen eines Ableitungsbaums, die durch Anwendung der Produktion p an einem Knoten des Ableitungsbaums erzeugt wurden. Demgegenüber beschreiben die Außenkanten des Auswertungsgraphen solche Abhängigkeiten, die durch Anwendung von Produktionen auf Knoten im Kontext von p erzeugt wurden.

Beispiel 4.15 (Fortsetzung von 3.5)
In Abbildung 4.4 sind die Auswertungsgraphen für MAG_{10} dargestellt.

■

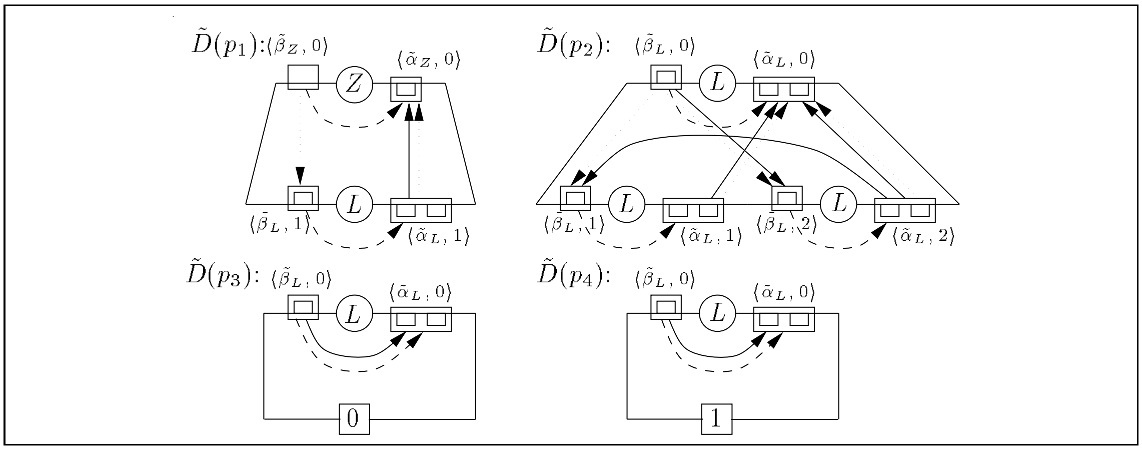


Abbildung 4.4: Auswertungsgraphen für MAG_{10}

Reps und Teitelbaum implementieren eine Variante der Konstruktion von Auswertungsgraphen in einem Algorithmus *Condense*, bei der die nach oben und unten erweiterten Abhängigkeitsgraphen einer Produktion p unmittelbar manipuliert werden. Dazu wählt der Algorithmus *Condense* aus jeder totalen Anordnung von Außenattributvorkommen einen Repräsentanten. Alle Kanten von und zu den Außenattributvorkommen, die in derselben Anordnung zusammengefaßt sind, werden auf den gewählten Repräsentanten der Anordnung „verbogen".[2]

Jede topologische Sortierung des Auswertungsgraphen erzeugt eine totale Ordnung auf den Anordnungen der Attributvorkommen, die für die Auswahl von Attributinstanzen

[2] vergleiche [RT89, Seite263f.]

von einem Auswerter benutzt werden kann. Bei Berücksichtigung des bereits gesammelten Wissens über die Abhängigkeiten zwischen den Attributvorkommen der Produktionen lassen sich spezielle topologische Sortierungen angeben, die effizienter konstruiert werden können als eine allgemeine topologische Sortierung. Diese speziellen topologischen Sortierungen werden in Analogie zu den Auswertungsordnungen der Nichtterminalsymbole als *Auswertungsordnungen der Produktionen* bezeichnet. Die Auswertungsordnungen der Produktionen haben die interessante Eigenschaft, daß auf jede Anordnung von Innenattributvorkommen stets eine Anordnung von Außenattributvorkommen folgt. Hinter dieser Struktur verbirgt sich wieder die beabsichtigte Auswertungsstrategie: Sind in einem Zustand des Auswerters die Werte aller inheriten beziehungsweise aller synthetischen Attributinstanzen eines Knotens x berechnet worden, die im gegenwärtigen Zustand berechnet werden konnten, wird die Traversierung des Ableitungsbaums mit dem Besuch von x beziehungsweise mit dem Besuch von $x.$-1 fortgesetzt.

Definition 4.16 (Auswertungsordnung einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine geordnete Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $p : X_0 \to w_0 X_1 w_1 \ldots X_n w_n \in P$. Sei $\tilde{D}(p) = (\tilde{A}(p), \tilde{E}(p))$ der Auswertungsgraph von p. Eine *aus $\tilde{D}(p)$ abgeleitete Auswertungsordnung von p*, bezeichnet mit T_p, ist eine topologische Sortierung

$$\tilde{o}_1 \tilde{\imath}_1 \ldots \tilde{o}_l \tilde{\imath}_l, \quad 2l = |\, \tilde{A}(p)\,|,$$

aller Anordnungen von Attributvorkommen von p, wobei für alle $j \in \{1, \ldots, l\}$ gilt: $\tilde{o}_j$ ist eine Anordnung von Außenattributvorkommen, und $\tilde{\imath}_j$ ist eine Anordnung von Innenattributvorkommen, und für alle $i \in \{1, \ldots, l-1\}$ ist $(\tilde{\imath}_i, \tilde{o}_{i+1})$ eine Außenkante von $\tilde{D}(p)$.

□

Beispiel 4.17 (Fortsetzung von 3.5)
Aus den Auswertungsgraphen der Produktionen werden folgende Auswertungsordnungen für p_1, p_2, p_3 und p_4 konstruiert:

$$\begin{aligned}
T_{p_1} &= \mathit{topsort}(\tilde{D}(p_1)) = \langle \tilde{\beta}_Z, 0\rangle \langle \tilde{\beta}_L, 1\rangle \langle \tilde{\alpha}_L, 1\rangle \langle \tilde{\alpha}_Z, 0\rangle \\
T_{p_2} &= \mathit{topsort}(\tilde{D}(p_2)) = \langle \tilde{\beta}_L, 0\rangle \langle \tilde{\beta}_L, 2\rangle \langle \tilde{\alpha}_L, 2\rangle \langle \tilde{\beta}_L, 1\rangle \langle \tilde{\alpha}_L, 1\rangle \langle \tilde{\alpha}_L, 0\rangle \\
T_{p_3} &= \mathit{topsort}(\tilde{D}(p_3)) = \langle \tilde{\beta}_L, 0\rangle \langle \tilde{\alpha}_L, 0\rangle \\
T_{p_4} &= \mathit{topsort}(\tilde{D}(p_4)) = \langle \tilde{\beta}_L, 0\rangle \langle \tilde{\alpha}_L, 0\rangle
\end{aligned}$$

□

Die eingangs erwähnten Pläne sind Kodierungen der gefundenen Auswertungsordnungen der Produktionen in Form von Steuerbefehlen. Aus der linearen Struktur der Auswertungsordnung T_p einer Produktion p resultiert eine lineare Struktur des Plans von p.

Entsprechend der beabsichtigten Auswertungsstrategie werden zwei Gruppen von Befehlen zur Kodierung benötigt:

1. Befehle zur Steuerung der Traversierung und
2. Befehle zur Berechnung der Werte der Attributinstanzen.

Ausgehend von einem Knoten x sind zwei verschiedene Bewegungen im Ableitungsbaum möglich: der Wechsel zum j-ten Nachfolgerknoten $x.j$ von x, kodiert durch einen Befehl $Visit(j)$, und der Wechsel zu dem Vorgängerknoten $x.$-1 von x, kodiert durch den Befehl $Suspend$. Beide Befehle werden mit einem weiteren Parameter ausgestattet, der angibt, zum wievielten Male der betreffende Knotenwechsel erfolgt: Es gibt $k(X_j)$, $1 \leq j \leq n$, $Visit(j)$-Befehle, falls der Knoten x n Nachfolger hat und der Knoten $x.j$ mit X_j beschriftet ist, und es gibt $k(X_0)$ $Suspend$-Befehle, falls x mit X_0 beschriftet ist. Ein Knotenwechsel muß immer dann erfolgen, wenn für die Berechnung der Werte weiterer Attributinstanzen zusätzlich Werte von Instanzen benötigt werden, die mit Außenattributvorkommen der Produktion $prod_t(x)$ assoziiert sind, da deren Werte von Attributinstanzen im Kontext von p abhängig sind.

Die Berechnung des Wertes einer Attributinstanz $\langle a, x.j\rangle$ wird mit dem Befehl $Eval(\langle a, j\rangle)$ veranlaßt, falls $\langle \alpha, j\rangle$ ein Innenattributvorkommen der Produktion ist, die an x angewendet wurde.

Definition 4.18 (Plan einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine geordnete Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $p : X_0 \to w_0X_1w_1 \dots X_nw_n \in P$. Sei $T_p = \tilde{o}_1\tilde{\imath}_1 \dots \tilde{o}_l\tilde{\imath}_l = \tilde{h}_1\tilde{h}_2 \dots \tilde{h}_{2l-1}\tilde{h}_{2l}$, $2l = |\,\tilde{A}(p)\,|$, Auswertungsordnung von p.

Ein *Plan von p*, bezeichnet mit $Plan(p)$, ist eine Sequenz von Instruktionen

$$Plan(p) = Instruktion_1; \dots ; Instruktion_{2l}; Suspend(k(X_0)),$$

wobei für alle $x \in \{1, \dots, 2l\}$ gilt:

$$\begin{aligned}
Instruktion_x = & Visit(j, i), \\
& \quad \text{falls } \tilde{h}_x = \langle \tilde{\alpha}_i, j\rangle, 1 \leq j \leq n, 1 \leq i \leq k(X_j), \\
& \quad \text{und } \langle \tilde{\alpha}_i, j\rangle \text{ ist Anordnung von Außenattributvorkommen} \\
= & Suspend(i-1), \\
& \quad \text{falls } \tilde{h}_x = \langle \tilde{\beta}_i, 0\rangle, 1 \leq i \leq k(X_0), \\
& \quad \text{und } \langle \tilde{\beta}_i, 0\rangle \text{ ist Anordnung von Außenattributvorkommen} \\
= & Eval(\langle c_1, j\rangle); \dots ; Eval(\langle c_r, j\rangle), \\
& \quad \text{falls } \tilde{h}_x = \langle \tilde{\gamma}_i, j\rangle, 0 \leq j \leq n, 1 \leq i \leq k(X_j) \\
& \quad \text{und } \langle \tilde{\gamma}_i, j\rangle \text{ ist Anordnung von Innenattributvorkommen} \\
& \quad \text{mit } \langle \tilde{\gamma}_i, j\rangle = \langle c_1, j\rangle; \dots ; \langle c_r, j\rangle, r \geq 0
\end{aligned}$$

□

In Definition 4.18 wird eine Instruktion $Suspend(0)$ als Kodierung von $\tilde{o}_1$ definiert. Die Abarbeitung einer $Suspend(0)$-Instruktion bedeutet, daß der Auswerter einen Teilbaum des Ableitungsbaums zum *nullten* mal verlassen soll. Diese Instruktion trägt nicht zur Steuerung der Traversierung bei und wird deshalb vernachlässigt. Anordnungen $\tilde{\imath}$ von Innenattributvorkommen, für die $\tilde{\imath}_i = \varepsilon$ gilt, werden nach Definition 4.18 durch eine $Eval(\varepsilon)$-Instruktion kodiert. Diese Instruktion trägt nicht zur Berechnung eines Wertes einer Attributinstanz bei und wird deshalb vernachlässigt.

Beispiel 4.19 (Fortsetzung von 3.5)
Aus den Auswertungsordnungen $T_{p_1}, T_{p_2}, T_{p_3}$ und T_{p_4} werden folgende Pläne konstruiert:

$$\begin{aligned}
Plan(p_1) &= Eval(\langle v,1\rangle); Visit(1,1); Eval(\langle \alpha_0,0\rangle); Suspend(1) \\
Plan(p_2) &= Eval(\langle v,2\rangle); Visit(2,1); Eval(\langle v,1\rangle); Visit(1,1); \\
&\quad Eval(\langle w,0\rangle); Eval(\langle u,0\rangle); Suspend(1) \\
Plan(p_3) &= Eval(\langle w,0\rangle); Eval(\langle u,0\rangle); Suspend(1) \\
Plan(p_4) &= Eval(\langle w,0\rangle); Eval(\langle u,0\rangle); Suspend(1)
\end{aligned}$$

■

Durch den Algorithmus 4.20 werden die Pläne ohne explizite Konstruktion der Auswertungsordnungen der Produktionen erzeugt. Da Definition 4.18 eine Substitution der Komponenten der Auswertungsordnung einer Produktion spezifiziert, ist die unmittelbare Erzeugung des zugehörigen Plans während der topologischen Sortierung des Auswertungsgraphen einer Produktion möglich.

Der Algorithmus 4.20 generiert zusätzlich eine Funktion $MapToPlanIndex$, mit deren Hilfe die Position derjenigen Instruktion bestimmt werden kann, die unmittelbar nach einem Knotenwechsel auszuführen ist. Mit einem Knotenwechsel ist immer ein Wechsel des abzuarbeitenden Plans verbunden. Wurde der Knotenwechsel durch einen $Visit(j,i)$-Befehl bewirkt, so ist der Plan der Produktion, die am j-ten Nachfolgerknoten angewendet wurde mit der Instruktion fortzusetzen, die dem $Suspend(i-1)$-Aufruf folgt. Wurde der Knotenwechsel durch einen $Suspend(i)$-Aufruf erzwungen, so ist der Plan der Produktion, die am Vorgängerknoten angewendet wurde, mit der Instruktion fortzusetzen, die dem i-ten Visitaufruf zu dem Nachfolgerknoten folgt.

Die Pläne zweier Produktionen $p = prod_t(x_p)$ und $q = prod_t(x_q)$, die an zwei Knoten x_p und x_q eines Ableitungsbaums angewendet wurden, sind dann wie folgt miteinander verzahnt:

$$Plan(p):\ \ldots; Visit(j,i); Instruktion_{pos_{Plan(p)}(Visit(j,i))+1}; \ldots$$

$$Plan(q):\ \ldots; Suspend(i-1); Instruktion_{pos_{Plan(q)}(Suspend(i-1))+1}; \ldots; Suspend(i); \ldots$$

Die Positionen $pos_{Plan(p)}(Visit(i,j)) + 1$ und $pos_{Plan(q)}(Suspend(i-1)) + 1$ lassen sich während der Konstruktion der Pläne protokollieren, sodaß sie nicht während des Auswerterlaufs „ausgezählt“ werden müssen. Das benötigte Protokoll wird durch die Funktion $MapToPlanIndex$ wie folgt angelegt: Sei $p : X_0 \to w_0X_1w_1 \ldots X_nw_n$, $n \geq 0$ die Produktion, für die ein Plan erstellt werden soll. Hat der Plan die Länge l und wird der Plan um einen $Visit(j,i)$-Aufruf verlängert, dann wird in die $MapToPlanIndex$-Funktion für das Tupel (p,j,i) der Wert $l+2$ eingetragen. Wird hingegen der Plan um einen $Suspend(i)$-Aufruf verlängert, dann muß in die $MapToPlanIndex$-Funktion für das Tupel $(p,0,i)$ der Wert $l+2$ eingetragen werden, wobei die 0 im zweiten Argument den Wechsel zum Vorgängerknoten kodiert. Ein Eintrag für $(p,0,i)$ ist jedoch nur sinnvoll, falls $i < k(X_0)$ gilt, da auf ein $Suspend(k(X_0))$ keine Instruktionen folgen.

Der Plan einer Produktion wird als Feld implementiert, um den Zugriff auf eine Instruktion über ihre Position im gesamten Plan zu realisieren. Die Pläne aller Produktionen werden wiederum in einem Feld zusammengefaßt, so daß der Zugriff auf einen Plan über die Nummer einer Produktion erfolgt. Der Ausdruck $Plan[p][x]$ beschreibt demzufolge die x-te Instruktion des Plans der Produktion p.

Algorithmus 4.20 (Aufstellung von Plänen)

Eingabe : für jedes $p \in P$ den Auswertungsgraph $\tilde{D}(p) = (\tilde{A}(p), \tilde{E}(p))$
Ausgabe : für jedes $p \in P$ einen Plan $Plan[p]$
mit der zugehörigen $MapToPlanIndex$-Funktion
Variable : $\$\tilde{A}$, $\$\tilde{E}$ zur Aufnahme eines $\tilde{D}(p)$
$\$PlanIndex$

```
foreach p ∈ P do begin
  $Ã := Ã(p); $Ẽ := Ẽ(p);
  $PlanIndex := 0;
  (* T_p[1] = ⟨β̃_1, 0⟩ wird durch ε ersetzt *)
  $Ã := $Ã \ {⟨β̃_1, 0⟩}; $Ẽ := $Ẽ⌈$Ã⌉;
  MapToPlanIndex(p, 0, 1) := 1;
  repeat
    foreach ⟨γ̃_i, j⟩ ∈ $Ã do begin
      if Es gibt kein ⟨δ̃, k⟩ ∈ $Ã mit (⟨δ̃, k⟩, ⟨γ̃_i, j⟩) ∈ $Ẽ then begin
          (* ⟨γ̃_i, j⟩ ist Anordnung von Innenattributvorkommen
          und Minimalelement von $Ẽ *)
          foreach ⟨c, j⟩ in ⟨γ̃_i, j⟩ do begin
            $PlanIndex := $PlanIndex + 1;
            Plan[p][$PlanIndex] := Eval(⟨c, j⟩);
          end
          $Ã := $Ã \ {⟨γ̃_i, j⟩};
          $PlanIndex := $PlanIndex + 1;
        if j = 0 then begin
          (* ⟨γ̃_i, j⟩ ist i-te Anordnung syn. Attributvorkommen bei X_0 *)
          Plan[p][$PlanIndex] := Suspend(i);
          if i < k(X_0) then begin
            (* Es folgen weitere Planinstruktionen *)
            $Ã := $Ã \ {⟨β̃_{i+1}, 0⟩}; $Ẽ := $Ẽ⌈$Ã⌉;
            (* Festlegen des Wiederaufnahmepunktes
              für (i + 1)-ten Visit-Aufruf vom Vorgängerknoten aus *)
            MapToPlanIndex(p, 0, i + 1) := $PlanIndex + 1;
          end
        end
        else begin (* j > 0 *)
          (* ⟨γ̃_i, j⟩ ist i-te Anordnung inh. Attributvorkommen bei X_j *)
          Plan[p][$PlanIndex] := Visit(j, i);
          $Ã := $Ã \ {⟨α̃_i, j⟩}; $Ẽ := $Ẽ⌈$Ã⌉;
          (* Festlegen des Wiederaufnahmepunktes
```

für i-tes $Suspend$ des j-ten Nachfolgerknotens *)
$MapToPlanIndex(p, j, i) := \$PlanIndex + 1;$
end ;
end ;
end ;
until $\$\tilde{A} = \emptyset$;
end

□

Beispiel 4.21 (Fortsetzung von 3.5)
Da jeder Ableitungsbaum in einem Durchlauf dekoriert werden kann, wird keiner der Pläne nach einem $Suspend$-Befehl wieder aufgenommen, sodaß die $MapToPlanIndex$-Funktion für die Pläne der Produktionen von MAG_{10} folgende Form hat:

$$MapToPlanIndex(p_1, 0, 1) = 1$$
$$MapToPlanIndex(p_1, 1, 1) = 3$$
$$MapToPlanIndex(p_2, 0, 1) = 1$$
$$MapToPlanIndex(p_2, 2, 1) = 3$$
$$MapToPlanIndex(p_2, 1, 1) = 5$$
$$MapToPlanIndex(p_3, 0, 1) = 1$$
$$MapToPlanIndex(p_4, 0, 1) = 1$$

■

Der Algorithmus zur iterativen Berechnung der zulässigen Termdekoration für geordnete Makroattributgrammatiken ist ein Interpreter für die Pläne der Produktionen.

Im Algorithmus 4.22 wird eine Funktion benutzt, die bisher noch nicht besprochen wurde. Die Funktion, die mit $NachfolgerNr$ bezeichnet wird, ist eine Funktion, die für einen Knoten x und seinen Vorgängerknoten $x.\text{-}1$ die Nummer i zurückliefert, falls $x = x.\text{-}1.i$, also x der i-te Nachfolger von $x.\text{-}1$ ist.

Algorithmus 4.22 (Iterativer Auswerter)
Eingabe : ein Ableitungsbaum $t \in DT(G_0)$
Ausgabe : eine zulässige Termdekoration $wert_t$ von t
Variable : x zur Aufnahme des zu Beginn eines Schleifendurchlaufs aktiven Knotens
$Planindex$ zur Identifizierung der nächsten zu befolgenden Planinstruktion
j zur Aufnahme der Nummer eines Nachfolgerknotens
r zur Aufnahme der Nummer eines Besuchs
$Instruktion$ zur Aufnahme einer Instruktion eines Plans

begin
$x := root(t);$
$PlanIndex := 1;$
loop

$Instruktion := Plan[prod_t(x)][PlanIndex]$;
if $Instruktion = Eval(\langle a, j\rangle)$ **then**
Sei $(\langle a, j\rangle(y_1, \ldots, y_m) = \zeta) \in RP(prod_t(x)), m \geq 0$
$wert_t(\langle a, x.j\rangle) := \Xi([\zeta]_x)$;
$PlanIndex := PlanIndex + 1$;
elsif $Instruktion = Visit(j, r)$ **then**
$PlanIndex := MapToPlanIndex(prod_t(x.j), 0, r)$;
$x := x.j$;
elsif $Instruktion = Suspend(r)$ **then**
if $x = root(t)$ **then** return **end** ;
$PlanIndex := MapToPlanIndex(prod_t(x.\text{-}1), NachfolgerNr(x, x.\text{-}1), r)$;
$x := x.\text{-}1$;
end
end
end

□

Beispiel 4.23 (Fortsetzung von 3.5)
Für den Ableitungsbaum t_{10} aus Abbildung 3.4 erzeugt der Algorithmus 4.22 die zulässige Termdekoration:

$$
\begin{array}{rclrcl}
wert_{t_{10}}(\langle v, 1\rangle)(y) &=& s(y) & wert_{t_{10}}(\langle v, 12\rangle)(y) &=& s(y) \\
wert_{t_{10}}(\langle v, 122\rangle)(y) &=& s(y) & wert_{t_{10}}(\langle w, 122\rangle)(y) &=& s(y) \\
wert_{t_{10}}(\langle u, 122\rangle)(y) &=& y & wert_{t_{10}}(\langle v, 121\rangle)(y) &=& s(s(y)) \\
wert_{t_{10}}(\langle v, 1212\rangle)(y) &=& s(s(y)) & wert_{t_{10}}(\langle w, 1212\rangle)(y) &=& s(s(y)) \\
wert_{t_{10}}(\langle u, 1212\rangle)(y) &=& s(s(y)) & wert_{t_{10}}(\langle v, 1211\rangle)(y) &=& s(s(s(s(y)))) \\
wert_{t_{10}}(\langle w, 1211\rangle)(y) &=& s(s(s(s(y)))) & wert_{t_{10}}(\langle u, 1211\rangle)(y) &=& y \\
wert_{t_{10}}(\langle w, 121\rangle)(y) &=& s(s(s(s(y)))) & wert_{t_{10}}(\langle u, 121\rangle)(y) &=& s(s(y)) \\
wert_{t_{10}}(\langle w, 12\rangle)(y) &=& s(s(s(s(y)))) & wert_{t_{10}}(\langle u, 12\rangle)(y) &=& s(s(y)) \\
wert_{t_{10}}(\langle v, 11\rangle)(y) &=& s^8(y) & wert_{t_{10}}(\langle w, 11\rangle)(y) &=& s^8(y) \\
wert_{t_{10}}(\langle u, 11\rangle)(y) &=& s^8(y) & wert_{t_{10}}(\langle w, 1\rangle)(y) &=& s^8(y) \\
wert_{t_{10}}(\langle u, 1\rangle)(y) &=& s^{10}(y) & wert_{t_{10}}(\langle \alpha_0, \varepsilon\rangle) &=& s^{10}(null)
\end{array}
$$

Aus der Anwendung der Terminterpretation gemäß Definition 3.24 auf $wert_{t_{10}}(\langle \alpha_0, \varepsilon\rangle)$ folgt sogleich $val_{t_{10}}(\langle \alpha_0, \varepsilon\rangle) = \hat{\varphi}(s^{10}(null)) = 10$.

■

Der Algorithmus 4.22 ist das Ergebnis der Modifikation eines iterativen Auswertungsalgorithmus für geordnete Attributgrammatiken, den Kastens in [Kas80, Abschnitt 6.4, Seite 250] vorschlägt. Kastens Algorithmus resultierte aus der von ihm durchgeführten Transformation eines rekursiven Auswerters geordneter Attributgrammatiken, der als Kellerautomat implementiert wird und der ebenfalls in [Kas80] diskutiert wird.

Die Darstellung des Algorithmus 4.22 basiert auf einer „modernisierten“ Variante von Kastens' Algorithmus, die von Reps und Teitelbaum in [RT89, Abbildung 12.2, Seite 253] angegeben wird. Die Modifikation von Reps und Teitelbaum betrifft ausschließlich die Form. Sie vermeidet die Darstellung des Auswertungsalgorithmus als endlicher Automat.

Kapitel 5

Inkrementeller Auswerter für geordnete Makroattributgrammatiken

Inkrementelle Auswertung ist von Reps und Teitelbaum für *Attributgrammatiken* untersucht worden. Ergebnis ihrer Arbeit ist ein Entwicklungssystem zur Konstruktion syntaxgesteuerter Editoren. Ihr System ist in der Lage, aus der Spezifikation einer geordneten Attributgrammatik einen syntaxgesteuerten Editor für die Sprache der kontextfreien Grammatik zu generieren, die der Attributgrammatik zugrundeliegt. Im Kern besitzt der generierte Editor einen inkrementellen Auswerter. Eine umfassende Darstellung des von ihnen aufgebauten Entwicklungssystems *Synthesizer Generator* findet sich in ihrer Monographie [RT89]. Aus der Monographie geht auch hervor, daß ihr Algorithmus zur inkrementellen Auswertung aus der Modifikation eines iterativen Auswerters für geordnete Attributgrammatiken resultiert, den Kastens in [Kas80] angegeben hat.

Kastens Algorithmus konnte, wie im Kapitel 4 der vorliegenden Arbeit beschrieben, an die Auswertung geordneter Makroattributgrammatiken angepaßt werden. Im Zusammenhang mit Beobachtung 3.18 zur Übertragbarkeit der Auswertungsstrategien für Attributgrammatiken auf Makroattributgrammatiken liegt es demzufolge nahe, die Überlegungen, die zur Entwicklung inkrementeller Auswerter führten, ebenfalls auf Makroattributgrammatiken anzuwenden.

5.1 Grundlegende Funktionsweise

Ausgangspunkt der Überlegungen ist eine neue Operation über Ableitungsbäumen, durch deren Anwendung ein Ableitungsbaum t in einen neuen Ableitungsbaum t' transformiert wird. Die Transformation von t in t' wird mit der Ersetzung eines Teilbaums $s \in sub(t)$ durch einen Baum s' erreicht. Einzige Voraussetzung für die Ersetzung von s durch s' ist dabei, daß die Wurzelknoten der beiden Bäume mit demselben Nichtterminalsymbol beschriftet sind: $label_s(root(s)) = label_{s'}(root(s'))$. Die erklärte Operation heißt *Teilbaumersetzung*. Eine solche Teilbaumersetzung ist in Abbildung 5.1 veranschaulicht.

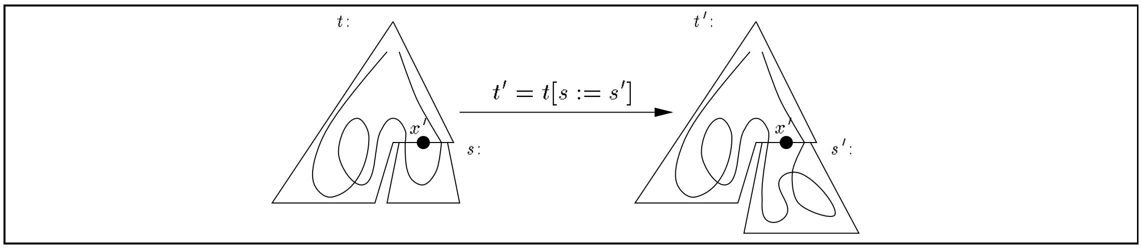


Abbildung 5.1: Ersetzung eines Teilbaums s eines Ableitungsbaums t

Durch wiederholte Ausführung solcher Teilbaumersetzungen läßt sich ausgehend von einem initialen Baum jeder Ableitungsbaum der zugrundeliegenden kontextfreien Grammatik erzeugen.

Verbunden mit der Ersetzung eines Teilbaums des Ableitungsbaums ist die Ersetzung eines Teilgraphen des Abhängigkeitsgraphen des Ableitungsbaums. In den seltensten Fällen werden die zulässige Termdekoration von t und die zulässige Termdekoration von t' übereinstimmen und t' muß (relativ zu t gesehen) neu dekoriert werden.

Die inkrementelle Auswertung vermeidet die vollständige Neuberechnung der zulässigen Termdekoration von t' und realisiert stattdessen eine *Anpassung* der zulässigen Termdekoration von t an die Änderungen, die zur Konstruktion von t' führten.

Die Idee der Anpassung der Termdekoration nach einer Teilbaumersetzung beruht auf der Tatsache, daß nach der Änderung des Ableitungsbaums t in t' eine Teilmenge $U(t') \subseteq A(t')$ der Attributinstanzen von dieser Änderung unberührt bleibt, in dem Sinne, daß für jede Attributinstanz $\langle a, x\rangle \in U(t')$ gilt: $wert_{t'}(\langle a, x\rangle) = wert_t(\langle a, x\rangle)$, also die Werte der Attributinstanzen in beiden Ableitungsbäumen übereinstimmen. Ist die zulässige Termdekoration von t bekannt, so kann eine zulässige Termdekoration für t' berechnet werden, indem

1. der neue Teilbaum s' zulässig dekoriert wird und

2. die Werte aller Attributinstanzen, die nicht in $U(t')$ liegen, neuberechnet werden.

Die Menge $A(t') \setminus U(t')$ der Attributinstanzen, deren Werte neu zu berechnen sind, wird als *betroffener Bereich* bezeichnet.

Da der betroffene Bereich verglichen mit der Menge aller Attributinstanzen des Ableitungsbaums t' von geringerer oder höchstens gleicher Mächtigkeit ist, ist die Anzahl der Berechnungen, die ein inkrementeller Auswerter ausführen muß, kleiner als bei der Berechnung der zulässigen Termdekoration des gesamten Ableitungsbaums t', höchstens aber genauso groß.

Ein Algorithmus, der eine inkrementelle Methode realisiert, benötigt demzufolge eine kürzere oder höchstens gleich lange Laufzeit gegenüber einem Algorithmus, der stets den gesamten Ableitungsbaum t' dekorieren muß.

Für die inkrementelle Auswertung muß ein Weg gefunden werden, den betroffenen Bereich einzugrenzen und zu verwalten, das heißt insbesondere, Start- und Endpunkt des Prozesses der Neuberechnung der zulässigen Termdekoration zu bestimmen.

5.2 Abgrenzung des betroffenen Bereichs

Der Knoten, an dem die Traversierung des Ableitungsbaums und die Berechnung der zulässigen Termdekoration nach der Änderung eines Ableitungsbaums wieder aufgenommen werden muß, ist der Wurzelknoten des Teilbaums, der ersetzt wurde. Diese Entscheidung ist aus folgendem Grunde korrekt:

Sei t ein Ableitungsbaum. Sei $x' \in node(t)$ der Knoten der Teilbaumersetzung. Sei $s = sub(t)$ mit $root(s) = x'$ der zu ersetzende Teilbaum von t. Sei t' der Ableitungsbaum, der entsteht, wenn im Ableitungsbaum t der Teilbaum s durch einen Baum s' ersetzt wird, für den $label_{s'}(root(s')) = label_t(x')$ gilt. Sei $X = label_{s'}(root(s'))$ dieses Nichtterminalsymbol. Sei $T_X = \tilde{\beta}_1\tilde{\alpha}_1 \ldots \tilde{\beta}_m\tilde{\alpha}_m$ die Auswertungsordnung des Nichtterminalsymbols $X \in N$. Sei $\langle b, x'\rangle \in A(t')$ die Instanz eines Attributs b aus $\tilde{\beta}_1$.

Aus der Annahme, daß sich der Wert von $\langle b, x'\rangle$ nach der Teilbaumersetzung geändert hat: $wert_{t'}(\langle b, x'\rangle) \neq wert_t(\langle b, x'\rangle)$, folgt, es muß eine Attributinstanz $\langle a, y\rangle$ im neuen Teilbaum s' geben, von der $\langle b, x'\rangle$ abhängig ist: $(\langle a, y\rangle, \langle b, x'\rangle) \in \rightarrow^+$, wobei $\rightarrow$ die Kantenrelation des Abhängigkeitsgraphen von t' ist. Dann muß es ein synthetisches Attribut $a' \in A(X)$ sowie eine Kante (a', b) im Graphen $is/si(X)$ geben. Die Kante (a', b) widerspricht jedoch der Auswertungsordnung T_X.

Demzufolge gilt für alle $b \in \tilde{\beta}_1$: $wert_{t'}(\langle b, x'\rangle) = wert_t(\langle b, x'\rangle)$.

Das bedeutet weiterhin, daß der iterative Auswerter, der die Traversierung am Wurzelknoten von t' beginnt, solange keine geänderten Werte von Makroattributinstanzen berechnen würde, bis der Auswerter den Knoten x' das erstemal nach der Teilbaumersetzung besucht.

Beginne der iterative Auswerter also die Traversierung am Wurzelknoten des neuen Teilbaums mit der Bearbeitung des Plans der Produktion, die am Wurzelknoten angewendet wurde. Es ist jedoch nicht notwendig, jede Attributinstanz neuzuberechnen, auf die der Auswerter während der Traversierung trifft, da möglicherweise Attributinstanzen in den wieder aufgenommenen Plänen auftauchen, die gar nicht von den Attributinstanzen des neuen Teilbaums abhängig sind.

Aus dieser Überlegung folgt eine erste Bedingung für die Neuberechnung einer Attributinstanz: Eine Attributinstanz des neuen Ableitungsbaums wird während des Auswerterlaufs höchstens dann in den betroffenen Bereich aufgenommen, wenn sie von einer Attributinstanz abhängig ist, die neu berechnet werden mußte. Praktischerweise wird diese Bedingung für die Implementation umgekehrt: Alle Attributinstanzen, die von einer neu berechneten Attributinstanz abhängig sind, werden in den betroffenen Bereich aufgenommen.

Die Menge der betroffenen Attributinstanzen induziert eine Menge von betroffenen Knoten: Besitzt ein Knoten x keine einzige Attributinstanz, die zum betroffenen Bereich gehört, so muß x auch nicht besucht werden, da aus einem solchen Besuch kein Beitrag für die Berechnung der neuen zulässigen Termdekoration zu erwarten ist. Die Menge betroffener Knoten wird zur Steuerung der Traversierung benutzt. Wenn in einem Plan der Wechsel zu einem Knoten gefordert wird, der nicht betroffen ist, so wird die auslösende $Visit$- beziehungsweise $Suspend$-Instruktion vom Auswerter ignoriert.

Wird über den „nackten“ Abhängigkeitsgraphen hinaus die Bedeutung des Begriffs „neu berechneter Wert“ auf den semantischen Bereich ausgedehnt, folgt eine zweite Bedingung für die Aufnahme von Attributinstanzen in den betroffenen Bereich. Möglicherweise ist der neu berechnete Wert einer Attributinstanz *nicht neu*, sondern mit dem alten Wert identisch. Dann können sich die Werte der Attributinstanzen, die von ersterer Instanz abhängig sind, ebenfalls nicht verändert haben.

Um die Verwaltung der Menge der betroffenen Attributinstanzen zu vereinfachen, werden zunächst *alle* Attributinstanzen eines Knotens als zum betroffenen Bereich gehörig betrachtet, wenn mindestens eine Attributinstanz des Knotens neu berechnet werden mußte. Der betroffene Bereich läßt sich dann durch die Menge betroffener Knoten repräsentieren. Im allgemeinen wird die Menge der Knoten eines Ableitungsbaums von geringerer Mächtigkeit als die Menge der Attributinstanzen des Ableitungsbaums sein, so daß die Verwaltung tatsächlich weniger aufwendig ist. Und die Menge der Knoten, die Attributinstanzen haben, die zum betroffenen Bereich gehören, muß ohnehin für die Steuerung der Traversierung bestimmt werden.

Die tatsächliche Anzahl der Attributinstanzen, die der Auswerter als betroffenen Bereich ansieht, ist dann abhängig von der Reihenfolge der Attributvorkommen in den Plänen der Produktionen. Das kommt daher, daß der Auswerter, wenn er nicht zwischen den Attributinstanzen eines Knotens unterscheiden kann, den Wert jeder Instanz neu berechnen und testen muß, die er während der Traversierung „sieht“, und diese Menge ist durch die Reihenfolge der Attributvorkommen in den Plänen vorgegeben.

Beispiel 5.1 (Fortsetzung von 3.5)
In Abbildung 5.2 ist der Ableitungsbaum t_{14} des Satzes 1110 dargestellt. Der Ableitungsbaum t_{14} entsteht, indem im Ableitungsbaum t_{10} aus Abbildung 3.4 der Teilbaum mit dem Wurzelknoten $root(t_{10}).1.2.1.1$ ersetzt wird. Der neue Teilbaum ist in der Abbildung 5.2 hervorgehoben. Aus dieser Ersetzung resultiert der betroffene Bereich

$$\{\langle w,1211\rangle,\langle w,121\rangle,\langle w,12\rangle,\langle w,1\rangle,\langle u,1211\rangle,\langle u,121\rangle,\langle u,12\rangle,\langle u,1\rangle,\langle \alpha_0,\varepsilon\rangle\},$$

der in der Abbildung durch $\circ$ oder $\bullet$ symbolisiert ist. Die Größe resultiert, wie oben erklärt, aus der Betrachtung *betroffener Knoten* in Verbindung mit der Reihenfolge der Attributvorkommen in den Plänen. Das ist der Grund, warum die Attributinstanz $\langle w,1\rangle$ ebenfalls dazugehört.

Wird der betroffene Bereich *nicht* durch eine Knotenmenge repräsentiert, enthält die Menge der betroffenen Attributinstanzen nur die Instanzen

$$\{\langle w,1211\rangle,\langle u,1211\rangle,\langle u,121\rangle,\langle u,12\rangle,\langle u,1\rangle,\langle \alpha_0,\varepsilon\rangle\},$$

■

Interessanterweise gehen Reps und Teitelbaum den umgekehrten Weg und schlagen als Optimierung des inkrementellen Auswerters die feinere Verwaltung einzelner Attributinstanzen vor. Dadurch erhöht sich zwar der Verwaltungsaufwand, jedoch verringert sich die Anzahl der Vergleiche von alten und neuen Werten der Attributinstanzen. Und jede mögliche Elimination eines Vergleichs von Attributwerten — was im Falle der Termdekoration den Vergleich von Termstrukturen bedeutet — hat eine wesentliche Verbesserung des Laufzeitverhaltens des Auswerters zur Folge. Dieser Gedanke wird im Kapitel 6, das sich mit der Optimierung des inkrementellen Auswerters auseinandersetzt, wieder aufgegriffen, doch vorerst soll die einfachere Verwaltung betroffener Knoten genügen.

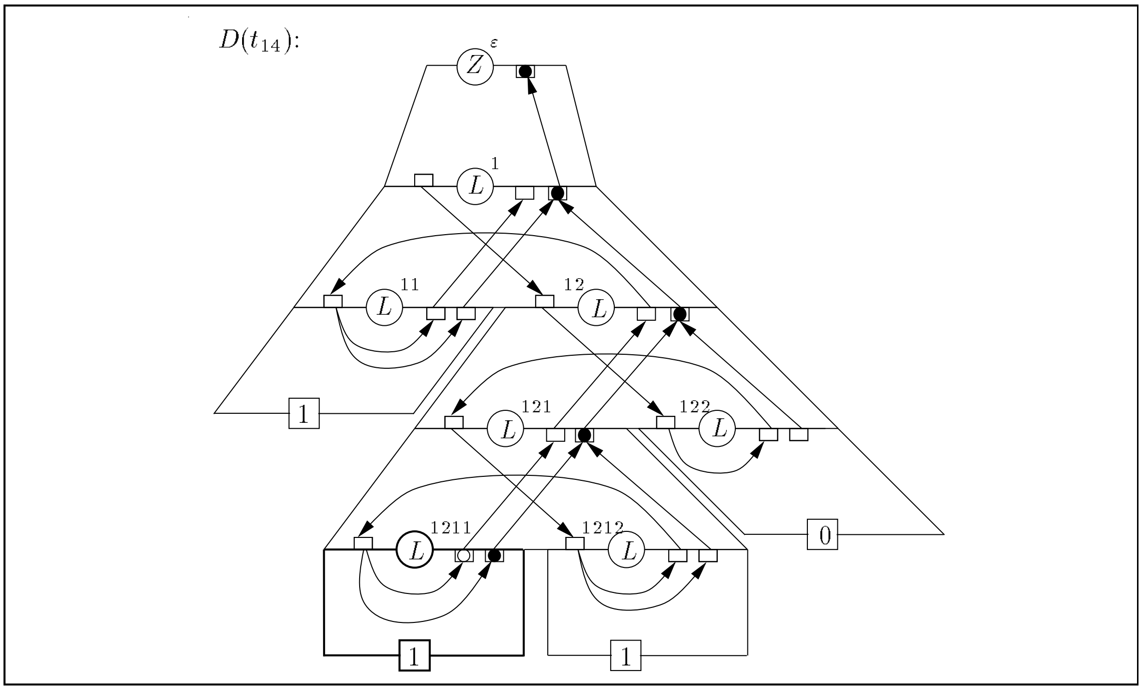


Abbildung 5.2: Der Abhängigkeitsgraph des Ableitungsbaums t_{14} des Satzes 1110

5.3 Algorithmus

Der Algorithmus zur inkrementellen Auswertung geordneter Makroattributgrammatiken ist eine Erweiterung des iterativen Auswerters für geordnete Makroattributgrammatiken aus dem Kapitel 4, für dessen Ablauf bereits Methoden zur Berechnung der Werte von Makroattributinstanzen entwickelt wurden.

Die Erweiterung des iterativen Auswerters um die dargestellten Ideen liefert einen inkrementellen Auswerter für geordnete Makroattributgrammatiken, der mit Algorithmus 5.2 implementiert wird.

Die eingerahmten Anweisungen sind die Anweisungen, um die der ursprünglich iterative Auswerter nach Algorithmus 4.22 erweitert wurde. Die eingefügten Kommentare dokumentieren die Implementation der Kontrolle des betroffenen Bereichs.

Algorithmus 5.2 (Inkrementeller Auswerter)
Eingabe : ein Ableitungsbaum $t \in DT(G_0)$
der Knoten x' der Teilbaumersetzung
die Termdekoration $wert_t$, die vor der Teilbaumersetzung zulässig war
für jede Produktion p $Plan(p)$ mit $MapToPlanIndex$-Funktion
Ausgabe : eine zulässige Termdekoration $wert_t$ von t
Variable : x zur Aufnahme des zu Beginn eines Schleifendurchlaufs aktiven Knotens
$Planindex$ zur Identifizierung der nächsten zu befolgenden Planinstruktion
j, j' zur Aufnahme der Nummer eines Nachfolgerknotens
r zur Aufnahme der Nummer eines Besuchs
$Instruktion$ zur Aufnahme einer Instruktion eines Plans

$Nachbar$ zur Aufnahme des Wurzelknotens eines Teilbaums,
der möglicherweise Attributinstanzen besitzt,
die von einer neu berechneten Attributinstanz abhängig sind
N_{neu} zur Aufnahme der Menge der neuen Knoten des Ableitungsbaums
$N_{betroffen}$ zur Aufnahme der Menge derjenigen Knoten,
die betroffene Attributinstanzen besitzen
$AlterWert \in T_\Phi(Y)$ zur Aufnahme eines bereits existierenden Wertes

begin
$\boxed{\text{Sei } s' \in sub(t) \text{ mit } root(s') = x' \text{ der neue Teilbaum;}}$
(* Die Menge der neuen Knoten ist dann: *)
$\boxed{N_{neu} := node(s') \setminus \{x'\};}$
(* Der betroffene Bereich umfaßt garantiert alle Instanzen im neuen Teilbaum: *)
$\boxed{N_{betroffen} := node(s');}$
(* Die Traversierung soll am Wurzelknoten x' des neuen Teilbaums beginnen: *)
$\boxed{x := x';}$
$PlanIndex := 1;$
loop
$Instruktion := Plan[prod_t(x)][PlanIndex];$
if $Instruktion = Eval(\langle a, j\rangle)$ **then**
(* Für jede Instanz im betroffenen Bereich, die nicht zum
neuen Teilbaum gehört, existiert bereits ein Wert, ... *)
$\boxed{\textbf{if } x.j \notin N_{neu} \textbf{ then } AlterWert := wert_t(\langle a, x.j\rangle); \textbf{ end };}$
Sei $(\langle a, j\rangle(y_1, \ldots, y_m) = \zeta) \in R(prod_t(x)), m \geq 0;$
$wert_t(\langle a, x.j\rangle) := \Xi([\zeta]_x);$
(* ... ein Wert, der mit dem neu berechneten verglichen werden muß: *)
$\boxed{\textbf{if } x.j \notin N_{neu} \textbf{ then}}$
(* Wenn tatsächlich ein neuer Wert vorliegt, ... *)
$\boxed{\textbf{if } wert_t(\langle a, x.j\rangle) \neq AlterWert \textbf{ then}}$
(* ... muß zusätzlich überprüft werden, ob von $\langle a, x.j\rangle$ weitere
Instanzen abhängen. Dazu muß als erstes gelöst werden,
WO die Abhängigkeiten zu suchen sind:
Fall 1: $\langle a, x.j\rangle$ ist inherit am j-ten Nachfolger
und mögliche Abhängigkeiten liegen „unterhalb“ von $x.j$ *)
$\boxed{\textbf{if } j \neq 0 \textbf{ then } Nachbar := x.j;}$
(* Fall 2: $\langle a, x.j\rangle$ ist synthetisch an x und mögliche Abhängigkeiten
liegen „oberhalb“ von x, wenn „oberhalb“ überhaupt existiert: *)
$\boxed{\textbf{elsif } x \neq root(t) \textbf{ then } Nachbar := x.\text{-}1;}$
(* Andere Möglichkeiten existieren nicht: *)
$\boxed{\textbf{else } Nachbar := NULL;}$
$\boxed{\textbf{end };}$
(* In den Fällen 1 und 2 wird $D(t)$ auf Abhängigkeiten überprüft ... *)
$\boxed{\textbf{if } Nachbar \neq NULL \textbf{ then}}$

(* ... und wenn von $\langle a, x.j\rangle$ abhängige Instanzen exitieren, ... *)
if Es gibt ein $\langle a', Nachbar.j'\rangle$ mit $(\langle a, x.j\rangle, \langle a', Nachbar.j'\rangle) \in\rightarrow$,
wobei $\rightarrow$ Kantenrelation von $D(t)$ ist **then**
(* ... muß der betroffene Bereich vergrößert werden: *)
$N_{betroffen} := N_{betroffen} \cup \{Nachbar\}$;
end **end** **end** **end**
$PlanIndex := PlanIndex + 1$
elsif $Instruktion = Visit(j, r)$ **then**
(* Der Zweck der Berechnung der Menge betroffener Knoten:
Besuch eines Nachfolgerknotens $x.j$ nur dann, wenn $x.j$ betroffen ist ... *)
if $x.j \in N_{betroffen}$ **then**
$PlanIndex := MapToPlanIndex(prod_t(x.j), 0, r)$;
$x := x.j$;
else
(* ... wenn nicht, wird der ganze Teilbaum einfach übergangen: *)
$PlanIndex := PlanIndex + 1$;
end ;
elsif $Instruktion = Suspend(r)$ **then**
if $x = root(t)$ **then** return **end** ;
(* Der Zweck der Berechnung der Menge betroffener Knoten:
Besuch eines Vorgängerknotens $x.$-1 nur dann, wenn $x.$-1 betroffen ist ... *)
if $x.\text{-}1 \in N_{betroffen}$ **then**
$PlanIndex := MapToPlanIndex(prod_t(x.\text{-}1), NachfolgerNr(x, x.\text{-}1), r)$;
$x := x.\text{-}1$;
(* ... wenn nicht, dann nicht: jedoch, wenn das das letzte Suspend war: *)
elsif $r = k(label_t(x))$ **then** return (* ist nichts mehr zu tun *);
else
$PlanIndex := PlanIndex + 1$;
end **end** **end** **end**

$\square$

Beispiel 5.3 (Fortsetzung von 3.5)
Der inkrementelle Auswerter berechnet die Werte der Attributinstanzen, die in Abbildung 5.2 mit $\circ$ markiert sind, und erkennt die Gleichheit zu den alten Werten. Der inkrementelle Auswerter berechnet die Werte der Attributinstanzen, die mit $\bullet$ markiert sind, und erkennt eine Änderung gegenüber den alten Werten.

Nachfolgend ist die neue Termdekoration $wert_{t_{14}}$ neben der alten Termdekoration $wert_{t_{10}}$ aus Beispiel 4.23 angegeben.

$$
\begin{array}{rlcrl}
wert_{t_{10}}(\langle v,1\rangle)(y) = & s(y) & & wert_{t_{14}}(\langle v,1\rangle)(y) = & s(y) \\
wert_{t_{10}}(\langle v,12\rangle)(y) = & s(y) & & wert_{t_{14}}(\langle v,12\rangle)(y) = & s(y) \\
wert_{t_{10}}(\langle v,122\rangle)(y) = & s(y) & & wert_{t_{14}}(\langle v,122\rangle)(y) = & s(y) \\
wert_{t_{10}}(\langle w,122\rangle)(y) = & s(y) & & wert_{t_{14}}(\langle w,122\rangle)(y) = & s(y) \\
wert_{t_{10}}(\langle u,122\rangle)(y) = & y & & wert_{t_{14}}(\langle u,122\rangle)(y) = & y \\
wert_{t_{10}}(\langle v,121\rangle)(y) = & s(s(y)) & & wert_{t_{14}}(\langle v,121\rangle)(y) = & s(s(y)) \\
wert_{t_{10}}(\langle v,1212\rangle)(y) = & s(s(y)) & & wert_{t_{14}}(\langle v,1212\rangle)(y) = & s(s(y)) \\
wert_{t_{10}}(\langle w,1212\rangle)(y) = & s(s(y)) & & wert_{t_{14}}(\langle w,1212\rangle)(y) = & s(s(y)) \\
wert_{t_{10}}(\langle u,1212\rangle)(y) = & s(s(y)) & & wert_{t_{14}}(\langle u,1212\rangle)(y) = & s(s(y)) \\
wert_{t_{10}}(\langle v,1211\rangle)(y) = & s(s(s(s(y)))) & & wert_{t_{14}}(\langle v,1211\rangle)(y) = & s(s(s(s(y)))) \\
wert_{t_{10}}(\langle w,1211\rangle)(y) = & s(s(s(s(y)))) & \circ & wert_{t_{14}}(\langle w,1211\rangle)(y) = & s(s(s(s(y)))) \\
wert_{t_{10}}(\langle u,1211\rangle)(y) = & y & \bullet & wert_{t_{14}}(\langle u,1211\rangle)(y) = & s(s(s(s(y)))) \\
wert_{t_{10}}(\langle w,121\rangle)(y) = & s(s(s(s(y)))) & \circ & wert_{t_{14}}(\langle w,121\rangle)(y) = & s(s(s(s(y)))) \\
wert_{t_{10}}(\langle u,121\rangle)(y) = & s(s(y)) & \bullet & wert_{t_{14}}(\langle u,121\rangle)(y) = & s(s(s(s(s(s(y)))))) \\
wert_{t_{10}}(\langle w,12\rangle)(y) = & s(s(s(s(y)))) & \circ & wert_{t_{14}}(\langle w,12\rangle)(y) = & s(s(s(s(y)))) \\
wert_{t_{10}}(\langle u,12\rangle)(y) = & s(s(y)) & \bullet & wert_{t_{14}}(\langle u,12\rangle)(y) = & s(s(s(s(s(s(y)))))) \\
wert_{t_{10}}(\langle v,11\rangle)(y) = & s^8(y) & & wert_{t_{14}}(\langle v,11\rangle)(y) = & s^8(y) \\
wert_{t_{10}}(\langle w,11\rangle)(y) = & s^8(y) & & wert_{t_{14}}(\langle w,11\rangle)(y) = & s^8(y) \\
wert_{t_{10}}(\langle u,11\rangle)(y) = & s^8(y) & & wert_{t_{14}}(\langle u,11\rangle)(y) = & s^8(y) \\
wert_{t_{10}}(\langle w,1\rangle)(y) = & s^8(y) & \circ & wert_{t_{14}}(\langle w,1\rangle)(y) = & s^8(y) \\
wert_{t_{10}}(\langle u,1\rangle)(y) = & s^{10}(y) & \bullet & wert_{t_{14}}(\langle u,1\rangle)(y) = & s^{14}(y) \\
wert_{t_{10}}(\langle \alpha_0,\varepsilon\rangle) = & s^{10}(null) & \bullet & wert_{t_{14}}(\langle \alpha_0,\varepsilon\rangle) = & s^{14}(null)
\end{array}
$$

Damit ist $val_{t_{14}}(\langle \alpha_0,\varepsilon\rangle) = \hat{\varphi}(s^{14}(null)) = 14$, und $(t_{14}, 14) \in \tau_{tv}(MAG_{10})$.

$\blacksquare$

Kapitel 6

Optimierung des inkrementellen Auswerters

Im Folgenden wird ausschließlich die Optimierung der Laufzeit des inkrementellen Auswerters diskutiert.

Eine Übersicht zu Techniken der Speicheroptimierung bei Attributauswertung ist in einer früheren Arbeit des Autors[1] zusammengestellt. Aus dieser Übersicht geht hervor, daß Optimierungen des Speicherbedarfs des inkrementellen Auswerters nur beschränkt realisierbar sind. Der Grund hierfür liegt in der Natur des inkrementellen Auswerters selbst. Da nach der Änderung eines Ableitungsbaums so wenige Attributinstanzen wie möglich erneut berechnet werden sollen, müssen diejenigen Attributinstanzen gespeichert werden, die für die Anpassung der Dekoration benötigt werden. Da jedoch *nicht* vorhergesagt werden kann, wann welcher Teilbaum eines Ableitungsbaums ersetzt wird, müssen prinzipiell *alle* Attributinstanzen eines Ableitungsbaums gespeichert werden.

Der inkrementelle Auswerter selbst ist eine laufzeitoptimierte Variante des iterativen Auswerters aus Kapitel 4, die auf die Anpassung einer zulässigen Dekoration eines Ableitungsbaums zugeschnitten ist. Die Verbesserung des Laufzeitverhaltens wurde durch eine optimierte Traversierung und eine optimierte Auswahl von Attributinstanzen, deren Werte berechnet werden müssen, erreicht.

Zwei neue Ansatzpunkte für die Optimierung bieten die beiden aufwendigsten Operationen, die der inkrementelle Auswerter ausführen muß:

1. die Ξ-Reduktion zur Berechnung der Termdekoration einer Attributinstanz und

2. der Vergleich des alten und des neuen Wertes einer Attributinstanz.

Ziel des vorliegenden Kapitels ist die Entwicklung einer geeigneten Repräsentation für die Termdekoration, die es erlaubt, die beiden Operationen mit geringeren Kosten auszuführen.

[1] siehe [Jok95]

6.1 Inkrementeller Auswerter vs Ξ-Reduktion

Der inkrementelle Auswerter zeigt eine Eigenschaft der Ξ-Reduktion, die in krassem Widerspruch zur Idee der inkrementellen Auswertung steht:

Beobachtung 6.1 (Ξ-Reduktion im inkrementellen Auswerter)
Jedesmal, wenn der Wert einer Attributinstanz $\langle a, x.j\rangle$ neu berechnet werden soll, wird *der gesamte Term* auf der rechten Seite $[\zeta]_x$ der semantischen Gleichung, die den Wert von $\langle a, x.j\rangle$ spezifiziert, *vom Wurzelknoten bis hin zu den Blattknoten* ausgewertet.

□

Zur Illustration von Beobachtung 6.1 diene folgendes Beispiel:

Beispiel 6.2
Seien $\langle a, 1\rangle, \langle a, 11\rangle$ Attributinstanzen eines Ableitungsbaums t. Die beiden Attributinstanzen seien durch folgende semantische Gleichung von t verknüpft:

$$\langle a, 1\rangle = f(\langle a, 11\rangle(\zeta_{11}), \zeta_2)$$

Gelte weiterhin folgende Ausschlußbedingung für die Teilterme ζ_{11} und ζ_2: Es existiert kein Knoten $n \in node(\zeta_{11})$ mit $label(\zeta_{11}, n) = \langle a, 11\rangle$, und es existiert kein Knoten $n' \in node(\zeta_2)$ mit $label(\zeta_2, n') = \langle a, 11\rangle$.

Sei $wert_t(\langle a, 11\rangle) = h(y)$. Dann folgt $wert_t(\langle a, 1\rangle) = f(h(t_{11}), t_2)$ mit $t_{11} = \Xi([\zeta_{11}])$ und $t_2 = \Xi([\zeta_2])$.

Führt die Ersetzung eines Teilbaums von t zu einem Ableitungsbaum t' mit dem geänderten Wert $wert_{t'}(\langle a, 11\rangle) = g(y)$, ändert sich ebenfalls $wert_{t'}(\langle a, 1\rangle)$, und es müssen folgende Schritte zur Neuberechnung ausgeführt werden:

$$\begin{aligned} wert_{t'}(\langle a, 1\rangle) &= \Xi(f(\langle a, 11\rangle([\zeta_{11}]), [\zeta_2])) \\ &= f(\Xi(\langle a, 11\rangle([\zeta_{11}])), \Xi([\zeta_2])) \\ &= f(wert_{t'}(\langle a, 11\rangle)[y := \Xi([\zeta_{11}])], \Xi([\zeta_2])) \end{aligned}$$

wegen $wert_{t'}(\langle a, 11\rangle) = g(y)$ ergibt sich:

$$= f(g(y)[y := \Xi([\zeta_{11}])], \Xi([\zeta_2]))$$

An dieser Stelle wird die Ξ-Reduktion auf den Teiltermen $[\zeta_{11}]$ und $[\zeta_2]$ erneut angestoßen, die wegen der Ausschlußbedingung jedoch wieder t_{11} und t_2 erzeugt: $\Xi([\zeta_{11}]) = t_{11}$ und $\Xi([\zeta_2]) = t_2$. Demzufolge gilt:

$$= f(g(t_{11}), t_2)$$

■

Um die wiederholte Berechnung von Teiltermen zu vermeiden, wird die Idee der inkrementellen Auswertung auf die Teilterme der rechten Seiten der semantischen Gleichungen ausgedehnt.

Dazu ist jedoch die Struktur der Abhängigkeitsgraphen zu grob.

Volle Abhängigkeitsgraphen

Die semantischen Regeln einer Produktion induzieren einen *vollen Abhängigkeitsgraphen einer Produktion*, der im Gegensatz zum einfachen Abhängigkeitsgraphen der Produktion gemäß Definition 3.4 die Struktur der Terme auf den rechten Seiten der Regeln *nicht* vernachlässigt.

Definition 6.3 (Voller Abhängigkeitsgraph einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik mit $G_0 = (N, \Sigma, Z, P)$. Sei $p \in P$ eine Produktion von G_0. Sei $A(p)$ die Menge der Attributvorkommen von p, und sei $R(p)$ die Menge der semantischen Regeln von p. Der *volle Abhängigkeitsgraph von* p, bezeichnet mit $\Delta(p)$, ist der Graph $\Delta(p) = (V_\Delta(p), E_\Delta(p))$, für den gilt:

- $$\begin{aligned} V_\Delta(p) = &\{\langle a,j\rangle.\varepsilon \mid \langle a,j\rangle \in A(p)\} \\ &\cup \{\langle a,j\rangle.\eta \mid (\langle a,j\rangle(y_1,\dots,y_m) = \zeta_{\langle a,j\rangle}) \in R(p), \eta \in node(\zeta_{\langle a,j\rangle}), \eta \neq \varepsilon\} \end{aligned}$$
- $$\begin{aligned} E_\Delta(p) = &\{(\langle a,j\rangle.\eta i, \langle a,j\rangle.\eta) \mid (\langle a,j\rangle(y_1,\dots,y_m) = \zeta_{\langle a,j\rangle}) \in R(p), \eta, \eta i \in node(\zeta_{\langle a,j\rangle}), \\ &\qquad label(\zeta_{\langle a,j\rangle}, \eta) \in (\Phi \cup A(p))^{(\kappa_1\dots\kappa_k,\kappa_0)}, k \geq 0, \kappa_0, \dots, \kappa_k \in \mathrm{K}, 1 \leq i \leq k\} \\ &\cup \{(\langle a',j'\rangle.\varepsilon, \langle a,j\rangle.\eta) \mid (\langle a,j\rangle(y_1,\dots,y_m) = \zeta_{\langle a,j\rangle}) \in R(p), \eta \in node(\zeta_{\langle a,j\rangle}), \\ &\qquad \langle a',j'\rangle \in au\beta en(p), label(\zeta_{\langle a,j\rangle}, \eta) = \langle a',j'\rangle\} \end{aligned}$$

Die Knotenmenge $V_\Delta(p)$ von $\Delta(p)$ heißt *Menge der Termsegmentvorkommen.*

□

Beispiel 6.4 (Fortsetzung von 3.5)
Der volle Abhängigkeitsgraph $\Delta(p_1)$ der Produktion $p_1 : Z \to L$ der Makroattributgrammatik M_{10} ist wegen

$$\begin{aligned} A(p_1) &= \{\langle w,1\rangle, \langle u,1\rangle, \langle \alpha_0,0\rangle, \langle v,1\rangle\} \\ \text{und } R(p_1) &= \{\langle \alpha_0,0\rangle = \langle u,1\rangle(null), \langle v,1\rangle = s(y)\} \end{aligned}$$

der Graph $\Delta(p_1) = (V_\Delta(p_1), E_\Delta(p_1))$

$$\begin{aligned} \text{mit } V_\Delta(p_1) = &\{\langle w,1\rangle.\varepsilon, \langle u,1\rangle.\varepsilon, \langle \alpha_0,0\rangle.\varepsilon, \langle v,1\rangle.\varepsilon\} \\ &\cup \{\langle \alpha_0,0\rangle.1, \langle v,1\rangle.1\} \\ \text{und } E_\Delta(p_1) = &\{(\langle \alpha_0,0\rangle.1, \langle \alpha_0,0\rangle.\varepsilon), (\langle v,1\rangle.1, \langle v,1\rangle.\varepsilon)\} \\ &\cup \{(\langle u,1\rangle.\varepsilon, \langle \alpha_0,0\rangle.\varepsilon)\} \end{aligned}$$

Den Graph $\Delta(p_1)$ sowie die vollen Abhängigkeitsgraphen $\Delta(p_2)$, $\Delta(p_3)$ und $\Delta(p_4)$ der verbleibenden Produktionen veranschaulicht die Abbildung 6.1.

□

In analoger Weise induziert die Menge $R(t)$ der semantischen Gleichungen eines Ableitungsbaums t einen *vollen Abhängigkeitsgraphen von* t.

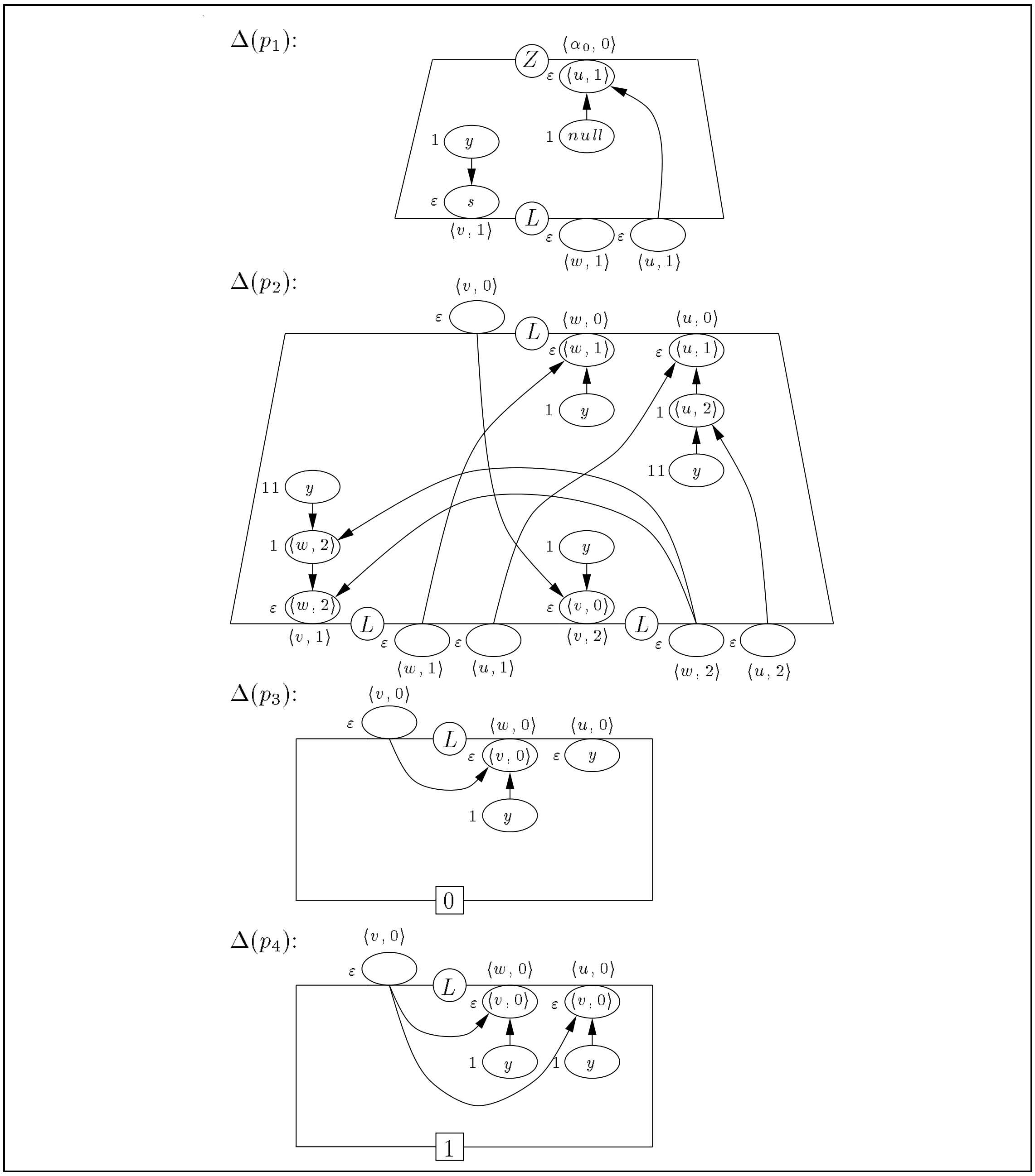


Abbildung 6.1: Volle Abhängigkeitsgraphen der Produktionen p_1, p_2, p_3 und p_4

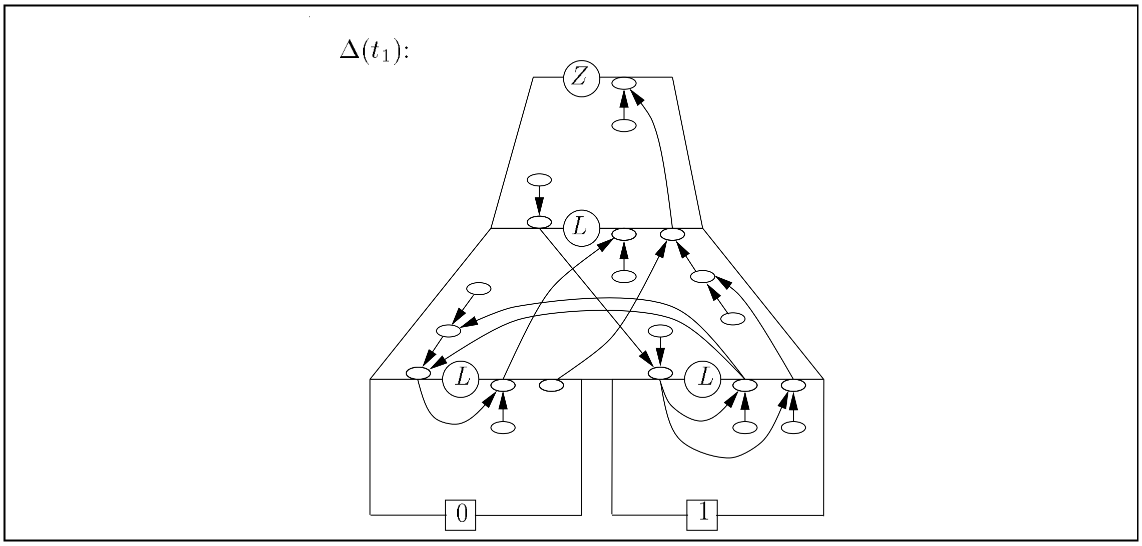


Abbildung 6.2: Voller Abhängigkeitsgraph des Ableitungsbaums t_{01}

Definition 6.5 (Voller Abhängigkeitsgraph eines Ableitungsbaums)
Sei $M = (G_0, D, B, R)$ eine Makroattributgrammatik mit $G_0 = (N, \Sigma, Z, P)$. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Sei $A(t)$ die Menge der Attributinstanzen von t. Sei $R(t)$ die Menge der semantischen Gleichungen von t. Der *volle Abhängigkeitsgraph von* t, bezeichnet mit $\Delta(t)$, ist der Graph $\Delta(t) = (V_\Delta(t), E_\Delta(t))$, für den gilt:

- $V_\Delta(t) = \{\langle a, x.j\rangle.\varepsilon \mid \langle a, x.j\rangle \in A(t)\}$
  $\cup\{\langle a, x.j\rangle.\eta \mid (\langle a, x.j\rangle(y_1, \ldots, y_m) = [\zeta_{\langle a,j\rangle}]_x) \in R(t), \eta \in node([\zeta_{\langle a,j\rangle}]_x), \eta \neq \varepsilon\}$
- $E_\Delta(t) = \{(\langle a, x.j\rangle.\eta i, \langle a, x.j\rangle.\eta) \mid (\langle a, x.j\rangle(y_1, \ldots, y_m) = [\zeta_{\langle a,j\rangle}]_x) \in R(t),$
  $\eta, \eta i \in node([\zeta_{\langle a,j\rangle}]_x), label([\zeta_{\langle a,j\rangle}]_x, \eta) \in (\Phi \cup A(t))^{(\kappa_1 \ldots \kappa_k, \kappa_0)},$
  $k \geq 0, \kappa_0, \ldots, \kappa_k \in \mathrm{K}, 1 \leq i \leq k\}$
  $\cup\{(\langle a', x.j'\rangle.\varepsilon, \langle a, x.j\rangle.\eta) \mid (\langle a, x.j\rangle(y_1, \ldots, y_m) = [\zeta_{\langle a,j\rangle}]_x) \in R(t),$
  $\eta \in node([\zeta_{\langle a,x.j\rangle}]_x), \langle a', j'\rangle \in A(t), label([\zeta_{\langle a,j\rangle}]_x, \eta) = \langle a', x.j'\rangle\}$

Die Knotenmenge $V_\Delta(t)$ von $\Delta(t)$ heißt *Menge der Termsegmentinstanzen.*

□

Beispiel 6.6 (Fortsetzung von 3.5)
Der volle Abhängigkeitsgraph des Ableitungsbaums t_{01} des Satzes 01 der Sprache der Beispielgrammatik M_{10} ist in Abbildung 6.2 dargestellt. ■

Anmerkung: Ohne Beweis sei bemerkt, daß die Ordnungseigenschaften, die die einfachen Abhängigkeitsgraphen besitzen, ebenfalls für die vollen Abhängigkeitsgraphen gelten. Das heißt, ist der Abhängigkeitsgraph eines Ableitungsbaums t *nichtzirkulär*, so ist der volle Abhängigkeitsgraph von t ebenfalls *nichtzirkulär*. Und weiter: Existiert für jede Produktion eine Auswertungsordnung über ihren Attributvorkommen, so ist ebenfalls eine

Auswertungsordnung über den Termsegmentvorkommen, die durch die rechten Seiten der semantischen Regeln der Produktion induziert werden, konstruierbar.

Die Betrachtung voller Abhängigkeitsgraphen führt zu einer Veränderung der Betrachtung der semantischen Regeln der Produktionen einer gegebenen Makroattributgrammatik.

Segmentierung semantischer Regeln und semantischer Gleichungen

Intuitiv wird jede semantische Regel an jedem Knoten des Terms ζ auf ihrer rechten Seite zerlegt und als eine *Menge von Termsegmentregeln* aufgefaßt. Jede Termsegmentregel hat die Eigenschaft, daß auf ihrer rechten Seite ein Term $\zeta' \in T_{\Phi \cup V_\Delta(p)}(Y)$ über Funktionssymbolen, Termsegmentvorkommen und Parametern vorkommt, der eine Höhe $height(\zeta') \in I\!N$ besitzt, für die gilt:

$$height(\zeta') = 1 \text{ oder } height(\zeta') = 2.$$

Definition 6.7 (Termsegmentregeln einer Produktion)
Sei $M = (G_0, D, B, R)$ eine Makroattributgrammatik mit $G_0 = (N, \Sigma, Z, P)$. Sei $p \in P$ eine Produktion von G_0. Sei $A(p)$ die Menge der Attributvorkommen von p. Sei $R(p)$ die Menge der semantischen Regeln von p. Die *Menge der Termsegmentregeln*, bezeichnet mit $R_\Delta(p)$, ist die Menge $\bigcup_{\langle a,j\rangle \in innen(p)} R_\Delta(\langle a,j\rangle)$, wobei für alle $\langle a,j\rangle \in innen(p)$ mit $(\langle a,j\rangle(y_1,\ldots,y_m) = \zeta_{\langle a,j\rangle}) \in R(p)$ gilt:

$$R_\Delta(\langle a,j\rangle) = Trans(\zeta_{\langle a,j\rangle.\varepsilon})$$

wobei für alle $\zeta_{\langle a,j\rangle} \in T_{\Phi \cup außen(p)}(\{y_1,\ldots,y_m\})$ und für alle $\eta \in node(\zeta_{\langle a,j\rangle})$ gilt:

$$\begin{aligned}
Trans(\zeta_{\langle a,j\rangle.\eta}) = & \{\langle a,j\rangle.\eta = y_i\}, \\
& \text{falls } \zeta_{\langle a,j\rangle.\eta} = y_i \text{ und } y_i \in \{y_1,\ldots,y_m\} \\
= & \{\langle a,j\rangle.\eta = f(\langle a,j\rangle.\eta 1,\ldots,\langle a,j\rangle.\eta k)\} \cup \bigcup_{i \in \{1,\ldots,k\}} Trans(\zeta_{\langle a,j\rangle.\eta i}), \\
& \text{falls } \zeta_{\langle a,j\rangle.\eta} \text{ die Form } f(\zeta_{\langle a,j\rangle.\eta 1},\ldots,\zeta_{\langle a,j\rangle.\eta k}) \text{ hat,} \\
& \text{und } f \in \Phi^{(\kappa_1\ldots\kappa_k,\kappa_0)}, k \geq 0, \kappa_0,\ldots,\kappa_k \in \mathrm{K}, \\
& \text{und } \zeta^{\kappa_1}_{\langle a,j\rangle.\eta 1},\ldots,\zeta^{\kappa_k}_{\langle a,j\rangle.\eta k} \in T_{\Phi \cup außen(p)}(\{y_1,\ldots,y_m\}) \\
= & \{\langle a,j\rangle.\eta = \langle a',j'\rangle(\langle a,j\rangle.\eta 1,\ldots,\langle a,j\rangle.\eta k')\} \cup \bigcup_{i \in \{1,\ldots,k'\}} Trans(\zeta_{\langle a,j\rangle.\eta i}), \\
& \text{falls } \zeta_{\langle a,j\rangle.\eta} \text{ die Form } \langle a',j'\rangle(\zeta_{\langle a,j\rangle.\eta 1},\ldots,\zeta_{\langle a,j\rangle.\eta k'}) \text{ hat,} \\
& \text{und } \langle a',j'\rangle \in außen(p)^{(\kappa_1\ldots\kappa_{k'},\kappa_0)}, k' \geq 0, \kappa_0,\ldots,\kappa_{k'} \in \mathrm{K}, \\
& \text{und } \zeta^{\kappa_1}_{\langle a,j\rangle.\eta 1},\ldots,\zeta^{\kappa_{k'}}_{\langle a,j\rangle.\eta k'} \in T_{\Phi \cup außen(p)}(\{y_1,\ldots,y_m\})
\end{aligned}$$

□

Beispiel 6.8 (Fortsetzung von 3.5)
Für die Produktionen der Beispielgrammatik M_{10} werden folgende Mengen $R_\Delta(p_1)$, $R_\Delta(p_2)$, $R_\Delta(p_3)$, $R_\Delta(p_4)$ gemäß Definition 6.7 konstruiert:

$$
\begin{aligned}
R_\Delta(p_1) = R_\Delta(\langle \alpha_0, 0\rangle) \cup R_\Delta(\langle v, 1\rangle) \text{ mit } & R_\Delta(\langle \alpha_0, 0\rangle) = \{\langle \alpha_0, 0\rangle.\varepsilon = \langle u, 1\rangle(\langle \alpha_0, 0\rangle.1), \\
& \qquad \langle \alpha_0, 0\rangle.1 = \mathit{null}\} \\
& R_\Delta(\langle v, 1\rangle) = \{\ \langle v, 1\rangle.\varepsilon = s(\langle v, 1\rangle.1), \\
& \qquad \langle v, 1\rangle.1 = y\} \\
R_\Delta(p_2) = R_\Delta(\langle v, 1\rangle) \cup R_\Delta(\langle v, 2\rangle) & \\
\cup R_\Delta(\langle w, 0\rangle) \cup R_\Delta(\langle u, 0\rangle) \text{ mit } & R_\Delta(\langle v, 1\rangle) = \{\ \langle v, 1\rangle.\varepsilon = \langle w, 2\rangle(\langle v, 1\rangle.1), \\
& \qquad \langle v, 1\rangle.1 = \langle w, 2\rangle(\langle v, 1\rangle.11), \\
& \qquad \langle v, 1\rangle.11 = y\} \\
& R_\Delta(\langle v, 2\rangle) = \{\ \langle v, 2\rangle.\varepsilon = \langle v, 0\rangle(\langle v, 2\rangle.1), \\
& \qquad \langle v, 2\rangle.1 = y\} \\
& R_\Delta(\langle w, 0\rangle) = \{\ \langle w, 0\rangle.\varepsilon = \langle w, 1\rangle(\langle w, 0\rangle.1), \\
& \qquad \langle w, 0\rangle.1 = y\} \\
& R_\Delta(\langle u, 0\rangle) = \{\ \langle u, 0\rangle.\varepsilon = \langle u, 1\rangle(\langle u, 0\rangle.1), \\
& \qquad \langle u, 0\rangle.1 = \langle u, 2\rangle(\langle u, 0\rangle.11), \\
& \qquad \langle u, 0\rangle.11 = y\} \\
R_\Delta(p_3) = R_\Delta(\langle w, 0\rangle) \cup R_\Delta(\langle u, 0\rangle) \text{ mit } & R_\Delta(\langle w, 0\rangle) = \{\ \langle w, 0\rangle.\varepsilon = \langle v, 0\rangle(\langle w, 0\rangle.1), \\
& \qquad \langle w, 0\rangle.1 = y\} \\
& R_\Delta(\langle u, 0\rangle) = \{\ \langle u, 0\rangle.\varepsilon = y\} \\
R_\Delta(p_4) = R_\Delta(\langle w, 0\rangle) \cup R_\Delta(\langle u, 0\rangle) \text{ mit } & R_\Delta(\langle w, 0\rangle) = \{\ \langle w, 0\rangle.\varepsilon = \langle v, 0\rangle(\langle w, 0\rangle.1), \\
& \qquad \langle w, 0\rangle.1 = y\} \\
& R_\Delta(\langle u, 0\rangle) = \{\ \langle u, 0\rangle.\varepsilon = \langle v, 0\rangle(\langle u, 0\rangle.1), \\
& \qquad \langle u, 0\rangle.1 = y\}
\end{aligned}
$$

■

In Analogie zu den Instanziierungen der semantischen Regeln einer Produktion p, den semantischen Gleichungen, heißt jede Instanziierung der Termsegmentregeln von p, die durch Anwendung von p an einem Knoten eines Ableitungsbaums t induziert wird, *Termsegmentgleichung von t*.

Definition 6.9 (Termsegmentgleichungen eines Ableitungsbaums)
Sei $M = (G_0, D, B, R)$ eine Makroattributgrammatik mit $G_0 = (N, \Sigma, Z, P)$. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Sei $A(t)$ die Menge der Attributinstanzen von t. Sei $R(t)$ die Menge der semantischen Gleichungen von t. Sei $x \in node(t)$ ein Knoten von t. Die *Menge der Termsegmentgleichungen*, bezeichnet mit $R_\Delta(t)$, ist die Menge $\bigcup_{\langle a, x.j\rangle \in A(t)} R_\Delta(\langle a, x.j\rangle)$, wobei für alle $\langle a, x.j\rangle \in A(t)$ mit $(\langle a, x.j\rangle(y_1, \ldots, y_m) = [\zeta_{\langle a,j\rangle}]_x) \in R(t)$ gilt:

$$R_\Delta(\langle a, x.j\rangle) = Trans([\zeta_{\langle a,j\rangle.\varepsilon}]_x)$$

wobei für alle $[\zeta_{\langle a,j\rangle}]_x \in T_{\Phi\cup A(t)}(\{y_1,\ldots,y_m\})$ und für alle $\eta \in node([\zeta_{\langle a,j\rangle}]_x)$ gilt:

$$\begin{aligned}
Trans([\zeta_{\langle a,j\rangle.\eta}]_x) =& \{\langle a,x.j\rangle.\eta = y_i\},\\
&\quad \text{falls } [\zeta_{\langle a,j\rangle.\eta}]_x = y_i \text{ und } y_i \in \{y_1,\ldots,y_m\}\\
=& \{\langle a,x.j\rangle.\eta = f(\langle a,x.j\rangle.\eta 1,\ldots,\langle a,x.j\rangle.\eta k)\}\\
\cup& \bigcup_{i\in\{1,\ldots,k\}} Trans([\zeta_{\langle a,j\rangle.\eta i}]_x),\\
&\quad \text{falls } [\zeta_{\langle a,j\rangle.\eta}]_x = f([\zeta_{\langle a,j\rangle.\eta 1}]_x,\ldots,[\zeta_{\langle a,x.j\rangle.\eta k}]_x),\\
&\quad \text{und } f \in \Phi^{(\kappa_1\ldots\kappa_k,\kappa_0)}, k \geq 0, \kappa_0,\ldots,\kappa_k \in \mathrm{K},\\
&\quad \text{und } [\zeta_{\langle a,j\rangle.\eta 1}]_x^{\kappa_1},\ldots,[\zeta_{\langle a,x.j\rangle.\eta k}]_x^{\kappa_k} \in T_{\Phi\cup A(t)}(\{y_1,\ldots,y_m\})\\
=& \{\langle a,x.j\rangle.\eta = \langle a',x.j'\rangle(\langle a,x.j\rangle.\eta 1,\ldots,\langle a,x.j\rangle.\eta k')\}\\
\cup& \bigcup_{i\in\{1,\ldots,k'\}} Trans([\zeta_{\langle a,j\rangle.\eta i}]_x),\\
&\quad \text{falls } [\zeta_{\langle a,j\rangle.\eta}]_x = \langle a',x.j'\rangle([\zeta_{\langle a,j\rangle.\eta 1}]_x,\ldots,[\zeta_{\langle a,j\rangle.\eta k'}]_x),\\
&\quad \text{und } \langle a',x.j'\rangle \in A(t)^{(\kappa_1\ldots\kappa_{k'},\kappa_0)}, k' \geq 0, \kappa_0,\ldots,\kappa_{k'} \in \mathrm{K},\\
&\quad \text{und } [\zeta_{\langle a,j\rangle.\eta 1}]_x^{\kappa_1},\ldots,[\zeta_{\langle a,x.j\rangle.\eta k'}]_x^{\kappa_{k'}} \in T_{\Phi\cup A(t)}(\{y_1,\ldots,y_m\})
\end{aligned}$$

□

Beispiel 6.10 (Fortsetzung von 3.5)
Die Menge $R_\Delta(t_{01})$ der Termsegmentgleichungen des Ableitungsbaums t_{01}, der in Abbildung 6.2 dargestellt ist, ist die Menge

$$\begin{aligned}
R_\Delta(t_{01}) = \{\; &\langle\alpha_0,\varepsilon\rangle.\varepsilon = \langle u,1\rangle(\langle\alpha_0,\varepsilon\rangle.1), && \langle\alpha_0,\varepsilon\rangle.1 = null,\\
&\langle v,1\rangle.\varepsilon = s(\langle v,1\rangle.1), && \langle v,1\rangle.1 = y,\\
&\langle v,11\rangle.\varepsilon = \langle w,12\rangle(\langle v,11\rangle.1), && \langle v,11\rangle.1 = \langle w,12\rangle(\langle v,11\rangle.11),\\
&\langle v,11\rangle.11 = y, && \langle v,12\rangle.\varepsilon = \langle v,1\rangle(\langle v,12\rangle.1),\\
&\langle v,12\rangle.1 = y, && \langle w,1\rangle.\varepsilon = \langle w,11\rangle(\langle w,1\rangle.1),\\
&\langle w,1\rangle.1 = y, && \langle u,1\rangle.\varepsilon = \langle u,11\rangle(\langle u,1\rangle.1),\\
&\langle u,1\rangle.1 = \langle u,12\rangle(\langle u,1\rangle.11), && \langle u,1\rangle.11 = y,\\
&\langle w,11\rangle.\varepsilon = \langle v,11\rangle(\langle w,11\rangle.1), && \langle w,11\rangle.1 = y,\\
&\langle u,11\rangle.\varepsilon = y, && \langle w,12\rangle.\varepsilon = \langle v,12\rangle(\langle w,12\rangle.1),\\
&\langle w,12\rangle.1 = y, && \langle u,12\rangle.\varepsilon = \langle v,12\rangle(\langle u,12\rangle.1),\\
&\langle u,12\rangle.1 = y\}
\end{aligned}$$

■

Die Definition der *Termsegmentdekoration eines Ableitungsbaums* ist eine Erweiterung der Termdekoration des Ableitungsbaums: Neben Termsegmentinstanzen, die mit den Attributinstanzen des Ableitungsbaums korrespondieren, muß ebenfalls allen übrigen Termsegmentinstanzen ein Wert zugeordnet werden:

Definition 6.11 (Termsegmentdekoration eines Ableitungsbaums)
Sei $MAG = (G_0, D, B, R)$ eine Makroattributgrammatik. Sei $t \in DT(G_0)$ ein Ableitungsbaum von G_0. Sei $V_\Delta(t)$ die Menge der Termsegmentinstanzen von t. Eine *Termsegmentdekoration von* t, bezeichnet mit $wert_t^\Delta$, ist eine Abbildung

$$wert_t^\Delta : V_\Delta(t) \to T_\Phi(Y)$$

wobei für alle $\langle a, x\rangle.\eta^{(\kappa_1\ldots\kappa_k,\kappa_0)} \in V_\Delta(t)$ gilt: $wert_t^\Delta(\langle a, x\rangle.\eta) \in T_\Phi(Y)^{\kappa_0}$.

□

Definition 6.12 (Zulässige Termsegmentdekoration)
Eine Termsegmentdekoration $wert_t^\Delta$ heißt genau dann *zulässig*, wenn für alle Termsegmentgleichungen

$$\langle a, x.j\rangle.\eta = [\zeta_{\langle a,j\rangle.\eta}]_x$$

mit $sort(\langle a, x.j\rangle.\eta) = (\kappa_1 \ldots \kappa_m, \kappa_0)$ gilt:

$$wert_t^\Delta(\langle a, x.j\rangle.\eta) = \Xi_\Delta([\zeta_{\langle a,j\rangle.\eta}]_x),$$

wobei die Abbildung $\Xi_\Delta : T_{\Phi\cup V_\Delta(t)}(Y) \to T_\Phi(Y)$ folgendermaßen definiert ist: Für alle $z \in T_{\Phi\cup V_\Delta(t)}(Y)$ ist

$$\begin{aligned}
\Xi_\Delta(z) =& z, \text{ falls } z \in Y\\
=& f(wert_t^\Delta(z_1), \ldots, wert_t^\Delta(z_k)),\\
& \text{falls } z = f(z_1, \ldots, z_k) \text{ und } f \in \Phi^{(\kappa_1\ldots\kappa_k,\kappa_0)}\\
& \text{mit } k \geq 0 \text{ und für alle } i \in \{1, \ldots, k\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } z_i \in V_\Delta(t)^{\kappa_i}\\
=& wert_t^\Delta(\langle a', x.j'\rangle.\varepsilon)[y'_1 := wert_t^\Delta(z_1), \ldots, y'_{k'} := wert_t^\Delta(z_{k'})],\\
& \text{falls } z = \langle a', x.j'\rangle(z_1, \ldots, z_{k'}) \text{ und } \langle a', x.j'\rangle \in V_\Delta(t)^{(\kappa_1\ldots\kappa_{k'},\kappa_0)}\\
& \text{mit } k' \geq 0 \text{ und für alle } i \in \{1, \ldots, k'\} \text{ ist } \kappa_i \in \mathrm{K} \text{ und } z_i \in V_\Delta(t)^{\kappa_i}
\end{aligned}$$

□

Das folgende Lemma 6.13 garantiert, daß die zulässige Termsegmentdekoration $wert_t^\Delta$ jedes Ableitungsbaums t der kontextfreien Grammatik einer gegebenen Makroattributgrammatik mit der zulässigen Termdekoration $wert_t$ an den Stellen identisch ist, an denen $wert_t$ definiert ist.

Lemma 6.13 (Korrektheit der zulässigen Termsegmentdekoration)
Sei $M = (G_0, D, B, R)$ mit $G_0 = (N, \Sigma, Z, P)$ und $M \in NZMAG$. Sei $t \in DT(G_0)$ und $x \in node(t)$. Für jedes $m \geq 0$, für alle $i \in \{1, \ldots, m\}$, für alle $\kappa_0, \kappa_i \in \mathrm{K}$ und für alle $\langle a, x\rangle^{(\kappa_1\ldots\kappa_m,\kappa_0)} \in A(t)$ und alle $\langle a, x\rangle.\varepsilon^{(\kappa_1\ldots\kappa_m,\kappa_0)} \in V_\Delta(t)$ gilt:

$$wert_t(\langle a, x\rangle) = wert_t^\Delta(\langle a, x\rangle.\varepsilon)$$

wobei $wert_t$ zulässige Termdekoration und $wert_t^\Delta$ zulässige Termsegmentdekoration sind.

□

Beweis
Lemma 6.13 folgt aus den Definitionen 3.22, 6.9 und 6.12 durch strukturelle Induktion über den Termaufbau der rechten Seiten $[\zeta]_x$ der *semantischen Gleichungen* (Definition 3.8).

■

Die Anwendung der Terminterpretation (Definition 3.24) auf die Werte, die den Termsegmentinstanzen durch die zulässige Termsegmentdekoration zugeordnet werden, erzeugt eine zulässige Dekoration val_t. Diese Tatsache folgt unmittelbar aus den Lemmata 3.26 und 6.13.

Insbesondere folgt aus der Gültigkeit von Lemma 6.13 für die Termsegmentinstanz, die mit der Instanz des Bedeutungsattributs korrespondiert, die Gültigkeit von

$$val_t(\langle \alpha_0, \varepsilon \rangle) = \hat{\varphi}(wert_t^\Delta(\langle \alpha_0, \varepsilon \rangle.\varepsilon)).$$

Damit gilt für die durch eine Makroattributgrammatik M beschriebene Übersetzung: Für jeden Ableitungsbaum t und seine zulässige Termsegmentdekoration $wert_t^\Delta$ ist

$$(t, \hat{\varphi}(wert_t^\Delta(\langle \alpha_0, \varepsilon \rangle.\varepsilon))) \in \tau_{tv}(MAG).$$

Beispiel 6.14 (Fortsetzung von 3.5)
Nachfolgend ist die Abbildung $wert_{t_{01}}^\Delta$ für diejenigen Termsegmentinstanzen angegeben, von denen die Termsegmentinstanz $\langle \alpha_0, \varepsilon \rangle.\varepsilon$ von t_{01} abhängig ist. Diese Instanzen und ihre Abhängigkeiten sind in Abbildung 6.2 leicht zu verfolgen.

$$\begin{aligned}
wert_{t_{01}}^\Delta(\langle v, 1 \rangle.1) &= \Xi_\Delta(y) \\
&= y \\
wert_{t_{01}}^\Delta(\langle v, 1 \rangle.\varepsilon) &= \Xi_\Delta(s(\langle v, 1 \rangle.1)) \\
&= s(wert_{t_{01}}^\Delta(\langle v, 1 \rangle.1)) \\
&= s(y) \\
wert_{t_{01}}^\Delta(\langle v, 12 \rangle.1) &= \Xi_\Delta(y) \\
&= y \\
wert_{t_{01}}^\Delta(\langle v, 12 \rangle.\varepsilon) &= \Xi_\Delta(\langle v, 1 \rangle(\langle v, 12 \rangle.1)) \\
&= wert_{t_{01}}^\Delta(\langle v, 1 \rangle.\varepsilon)[y := wert_{t_{01}}^\Delta(\langle v, 12 \rangle.1)] \\
&= s(y)[y := y] \\
&= s(y) \\
wert_{t_{01}}^\Delta(\langle u, 12 \rangle.1) &= \Xi_\Delta(y) \\
&= y \\
wert_{t_{01}}^\Delta(\langle u, 12 \rangle.\varepsilon) &= \Xi_\Delta(\langle v, 12 \rangle(\langle u, 12 \rangle.1)) \\
&= wert_{t_{01}}^\Delta(\langle v, 12 \rangle.\varepsilon)[y := wert_{t_{01}}^\Delta(\langle u, 12 \rangle.1)] \\
&= s(y)[y := y] \\
&= s(y)
\end{aligned}$$

$$\begin{aligned}
wert^{\Delta}_{t_{01}}(\langle u, 11\rangle.\varepsilon) &= \Xi_{\Delta}(y) \\
&= y \\
wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.11) &= \Xi_{\Delta}(y) \\
&= y \\
wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.1) &= \Xi_{\Delta}(\langle u, 12\rangle(\langle u, 1\rangle.11)) \\
&= wert^{\Delta}_{t_{01}}(\langle u, 12\rangle.\varepsilon)[y := wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.11)] \\
&= s(y)[y := y] \\
&= s(y) \\
wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.\varepsilon) &= \Xi_{\Delta}(\langle u, 11\rangle(\langle u, 1\rangle.1)) \\
&= wert^{\Delta}_{t_{01}}(\langle u, 11\rangle.\varepsilon)[y := wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.1)] \\
&= y[y := s(y)] \\
&= s(y) \\
wert^{\Delta}_{t_{01}}(\langle \alpha_0, \varepsilon\rangle.1) &= \Xi_{\Delta}(null) \\
&= null \\
wert^{\Delta}_{t_{01}}(\langle \alpha_0, \varepsilon\rangle.\varepsilon) &= \Xi_{\Delta}(\langle u, 1\rangle(\langle \alpha_0, \varepsilon\rangle.1)) \\
&= wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.\varepsilon)[y := wert^{\Delta}_{t_{01}}(\langle \alpha_0, \varepsilon\rangle.1)] \\
&= s(y)[y := null] \\
&= s(null)
\end{aligned}$$

Die Terminterpretation liefert sofort:

$$\begin{aligned}
val_{t_{01}}(\langle \alpha_0, \varepsilon\rangle) &= \hat{\varphi}(wert^{\Delta}_{t_{01}}(\langle \alpha_0, \varepsilon\rangle.\varepsilon)) \\
&= \hat{\varphi}(s(null)) \\
&= 1.
\end{aligned}$$

Also ist das Tupel $(t_{01}, 1)$ Element von $\tau_{tv}(M_{10})$.

■

6.2 Algorithmus

Der optimierte inkrementelle Auswerter berechnet für jeden Ableitungsbaum t einer gegebenen Makroattributgrammatik die zulässige Termsegmentdekoration von t. Die Reihenfolge, in der die Knoten von t zu besuchen und die Werte der Termsegmentinstanzen von t zu berechnen sind, wird durch Pläne festgelegt, die zur Segmentierung der semantischen Gleichungen eines Ableitungsbaums „passen" müssen.

Die Anpassung der Pläne der Produktionen geschieht intuitiv folgendermaßen: Für jedes Attributvorkommen $\langle a, j\rangle \in innen(p)$ einer Produktion p muß im Plan $Plan(p)$ der Produktion die $Eval\langle a, j\rangle$-Instruktion durch eine Sequenz von $Eval(\langle a, j\rangle.\eta)$-Instruktionen ersetzt werden, die der Kantenrelation des vollen Abhängigkeitsgraphen genügt:

Definition 6.15 (Voller Plan einer Produktion)
Sei $MAG = (G_0, D, B, R)$ eine geordnete Makroattributgrammatik, $G_0 = (N, \Sigma, Z, P)$. Sei $p : X_0 \rightarrow w_0X_1w_1 \ldots X_nw_n \in P$. Sei $\Delta(p) = (V_\Delta(p), E_\Delta(p))$ der volle Abhängigkeitsgraph von p. Sei $Plan(p) = Instruktion_1; \ldots ; Instruktion_l$, $l > 0$, der Plan von p gemäß Definition 4.18. Ein *voller Plan von p*, bezeichnet mit $Plan_\Delta(p)$, ist eine Sequenz von Instruktionen

$$Plan_\Delta(p) = I_1^\Delta; \ldots ; I_m^\Delta, \quad m \in I\!N_+,$$

wobei für alle $x \in \{1, \ldots , m\}$ gilt:

$$\begin{aligned}
I_x^\Delta =\ & Visit(j,i), \\
& \quad \text{falls } Instruktion_x = Visit(j,i), 1 \leq j \leq n, 1 \leq i \leq k(X_j) \\
=\ & Suspend(i), \\
& \quad \text{falls } Instruktion_x = Suspend(i), 1 \leq i \leq k(X_0) \\
=\ & Eval(\langle a,j\rangle.\eta_1); \ldots ; Eval(\langle a,j\rangle.\eta_r), \\
& \quad \text{falls } Instruktion_x = Eval(\langle a,j\rangle) \text{ und } \langle a,j\rangle.\eta_1, \ldots , \langle a,j\rangle.\eta_r = topsort(V,E), \\
& \quad \text{wobei } V = V_\Delta(p) \cap \{\langle a,j\rangle.\eta_1, \ldots , \langle a,j\rangle.\eta_r\} \\
& \quad \text{und } E = E_\Delta(p)\lceil\{\langle a,j\rangle.\eta_1, \ldots , \langle a,j\rangle.\eta_r\}\rceil
\end{aligned}$$

□

Beispiel 6.16 (Fortsetzung von 3.5)
Die Anwendung von Definition 6.15 auf die Pläne $Plan(p_1)$, $Plan(p_2)$, $Plan(p_3)$, $Plan(p_4)$, die in Beispiel 4.19 angegeben sind, liefert:

$$\begin{aligned}
Plan_\Delta(p_1) =\ & Eval(\langle v,1\rangle.1); Eval(\langle v,1\rangle.\varepsilon); Visit(1,1); \\
& Eval(\langle \alpha_0,0\rangle.1); Eval(\langle \alpha_0,0\rangle.\varepsilon); Suspend(1) \\
Plan_\Delta(p_2) =\ & Eval(\langle v,2\rangle.1); Eval(\langle v,2\rangle.\varepsilon); Visit(2,1); \\
& Eval(\langle v,1\rangle.11); Eval(\langle v,1\rangle.1); Eval(\langle v,1\rangle.\varepsilon); Visit(1,1); \\
& Eval(\langle w,0\rangle.1); Eval(\langle w,0\rangle.\varepsilon); \\
& Eval(\langle u,0\rangle.11); Eval(\langle u,0\rangle.1); Eval(\langle u,0\rangle.\varepsilon); Suspend(1) \\
Plan_\Delta(p_3) =\ & Eval(\langle w,0\rangle.1); Eval(\langle w,0\rangle.\varepsilon); Eval(\langle u,0\rangle.\varepsilon); Suspend(1) \\
Plan_\Delta(p_4) =\ & Eval(\langle w,0\rangle.1); Eval(\langle w,0\rangle.\varepsilon); \\
& Eval(\langle u,0\rangle.1); Eval(\langle u,0\rangle.\varepsilon); Suspend(1)
\end{aligned}$$

Die zu den obigen Plänen gehörende $MapToPlanIndex$-Funktion ist:

$$\begin{aligned}
& MapToPlanIndex(p_1,0,1) = 1, \quad MapToPlanIndex(p_1,1,1) = 4, \\
& MapToPlanIndex(p_2,0,1) = 1, \quad MapToPlanIndex(p_2,2,1) = 4, \\
& MapToPlanIndex(p_2,1,1) = 8, \\
& MapToPlanIndex(p_3,0,1) = 1, \quad MapToPlanIndex(p_4,0,1) = 1
\end{aligned}$$

■

Die Generierung der vollen Pläne nebst ihrer *MapToPlanIndex*-Funktion kann mit einem modifizierten Algorithmus 4.20 realisiert werden: Anstatt *eine Eval-Instruktion* für *ein* Attributvorkommen an den Plan anzuhängen, muß der modifizierte Algorithmus eine *Sequenz von Eval-Instruktionen* für eine topologische Sortierung von Termsegmentvorkommen generieren. Da die notwendigen Änderungen, die am Algorithmus 4.20 vorzunehmen wären, intuitiv klar erscheinen, wird der resultierende Algorithmus nicht explizit angegeben.

Mit der Information aus den vollen Plänen wird ein optimierter inkrementeller Auswerter gesteuert, der eine Neuberechnung der zulässigen Termsegmentdekoration jedes Ableitungsbaums t beginnend bei den „Blättern" des vollen Abhängigkeitsgraphen von t durchführt.

Die wichtige Modifikation gegenüber Algorithmus 5.2 resultiert aus der *Rücknahme* der Idee, den betroffenen Bereich durch die *Menge der Knoten* zu repräsentieren, die Instanzen besitzen, deren Werte sich geändert haben: Der betroffene Bereich, im nachstehenden Algorithmus mit $V_{\Delta,betroffen}$ bezeichnet, ist eine Menge von Termsegmentinstanzen, er wird *nicht* durch eine Menge betroffener Knoten repräsentiert.

Algorithmus 6.17 (Inkrementeller Auswerter auf vollen Abhängigkeitsgraphen)
Eingabe : ein Ableitungsbaum $t \in DT(G_0)$ mit $\Delta(t) = (V_\Delta(t), E_\Delta(t))$
der Knoten x' der Teilbaumersetzung
die Termsegmentdekoration $wert_t^\Delta$, die vor der Teilbaumersetzung zulässig war
für jede Produktion p $Plan_\Delta(p)$ mit $MapToPlanIndex$-Funktion
Ausgabe : eine zulässige Termsegmentdekoration $wert_t^\Delta$ von t
Variable : x zur Aufnahme des zu Beginn eines Schleifendurchlaufs aktiven Knotens
$Planindex$ zur Identifizierung der nächsten zu befolgenden Planinstruktion
j, j' zur Aufnahme der Nummer eines Nachfolgerknotens
r zur Aufnahme der Nummer eines Besuchs
$Instruktion$ zur Aufnahme einer Instruktion eines Plans
$Nachbar$ zur Aufnahme des Wurzelknotens eines Teilbaums,
der möglicherweise Termsegmentinstanzen besitzt,
die von einer neuberechneten Termsegmentinstanz abhängig sind
N_{neu} zur Aufnahme der Menge der Knoten des neuen Teilbaums
$N_{betroffen}$ zur Aufnahme der Menge derjenigen Knoten,
die betroffene Termsegmentinstanzen besitzen
$AlterWert \in T_\Phi(Y)$ zur Aufnahme eines bereits existierenden Wertes
$V_{\Delta,betroffen}$ zur Aufnahme der Menge der betroffenen Termsegmentinstanzen

begin
Sei $s' \in sub(t)$ mit $root(s') = x'$ der neue Teilbaum mit $\Delta(s') = (V_\Delta(s'), E_\Delta(s'))$;
$N_{neu} := node(s') \setminus \{x'\}$;
$N_{betroffen} := node(s')$;
$V_{\Delta,betroffen} := V_\Delta(s')$;
$x := x'$;
$PlanIndex := 1$;
loop

$Instruktion := Plan[prod_t(x)][PlanIndex]$;
if $Instruktion = Eval(\langle a, j\rangle.\eta)$ **then**
  if $\langle a, x.j\rangle.\eta \in V_{\Delta,betroffen}$ **then**
    if $x.j \notin N_{neu}$ **then** $AlterWert := wert_t^\Delta(\langle a, x.j\rangle.\eta)$; **end** ;
    Sei $(\langle a, j\rangle.\eta = \zeta_{\langle a,j\rangle.\eta}) \in R_\Delta(prod_t(x))$;
    $wert_t^\Delta(\langle a, x.j\rangle.\eta) := \Xi_\Delta([\zeta_{\langle a,j\rangle.\eta}]_x)$;
    if $x.j \notin N_{neu}$ **then**
      if $wert_t^\Delta(\langle a, x.j\rangle.\eta) \neq AlterWert$ **then**
        if $\eta \neq \varepsilon$ **then** $Nachbar := x$
        elsif $j \neq 0$ **then** $Nachbar := x.j$
        elsif $x \neq root(t)$ **then** $Nachbar := x.\text{-}1$
        else $Nachbar := NULL$
        end ;
        if $Nachbar \neq NULL$ **then**
          if Es gibt ein $\langle a', Nachbar.j'\rangle.\eta'$ mit
            $(\langle a, x.j\rangle.\eta, \langle a', Nachbar.j'\rangle.\eta') \in E_\Delta(t)$ **then**
            $N_{betroffen} := N_{betroffen} \cup \{Nachbar\}$;
            foreach $\langle a', Nachbar.j'\rangle.\eta'$ mit
                $(\langle a, x.j\rangle.\eta, \langle a', Nachbar.j'\rangle.\eta') \in E_\Delta(t)$ **do**
              $V_{\Delta,betroffen} := V_{\Delta,betroffen} \cup \{\langle a', Nachbar.j'\rangle.\eta'\}$;
  end end end end end end
  $PlanIndex := PlanIndex + 1$
elsif $Instruktion = Visit(j, r)$ **then**
  if $x.j \in N_{betroffen}$ **then**
    $PlanIndex := MapToPlanIndex(prod_t(x.j), 0, r)$;
    $x := x.j$;
  else
    $PlanIndex := PlanIndex + 1$;
  end ;
elsif $Instruktion = Suspend(r)$ **then**
  if $x = root(t)$ **then** return **end** ;
  if $x.\text{-}1 \in N_{betroffen}$ **then**
    $PlanIndex := MapToPlanIndex(prod_t(x.\text{-}1), NachfolgerNr(x, x.\text{-}1), r)$;
    $x := x.\text{-}1$;
  elsif $r = k(label_t(x))$ **then** return;
  else
    $PlanIndex := PlanIndex + 1$;
end end end end

□

Beispiel 6.18 (Fortsetzung von 3.5)
Der inkrementelle Auswerter paßt, nachdem der Ableitungsbaum t_{01} aus Abbildung 6.2 in den Ableitungsbaum t_{11} von Abbildung 6.3 transformiert wurde, die zulässige Termsegmentdekoration von t_{01} wie folgt an:

Die Werte der Termsegmentinstanzen $\langle w, 11\rangle.1$ und $\langle w, 11\rangle.\varepsilon$ werden neu berechnet. Jedoch stellt sich heraus, daß die neuen Werte mit den alten übereinstimmen (in Abbildung 6.3

durch $\circ$ symbolisiert). Die Menge der betroffenen Instanzen enthält außerdem die Instanzen

$$\langle u, 11\rangle.1, \langle u, 11\rangle.\varepsilon, \langle u, 1\rangle.\varepsilon, \langle \alpha_0, \varepsilon\rangle.\varepsilon,$$

die in Abbildung 6.3 mit $\bullet$ markiert sind.

$$\begin{aligned}
wert^{\Delta}_{t_{11}}(\langle u, 11\rangle.1) =& \Xi_{\Delta}(y)\\
=& y\\
\neq& wert^{\Delta}_{t_{01}}(\langle u, 11\rangle.1)\\
wert^{\Delta}_{t_{11}}(\langle u, 11\rangle.\varepsilon) =& \Xi_{\Delta}(\langle v, 11\rangle(\langle u, 11\rangle.1))\\
=& wert^{\Delta}_{t_{11}}(\langle v, 11\rangle.\varepsilon)[y := wert^{\Delta}_{t_{11}}(\langle u, 11\rangle.1)]\\
=& s(s(y))[y := y]\\
=& s(s(y))\\
\neq& wert^{\Delta}_{t_{01}}(\langle u, 11\rangle.\varepsilon)\\
wert^{\Delta}_{t_{11}}(\langle u, 1\rangle.\varepsilon) =& \Xi_{\Delta}(\langle u, 11\rangle(\langle u, 1\rangle.1))\\
=& wert^{\Delta}_{t_{11}}(\langle u, 11\rangle.\varepsilon)[y := wert^{\Delta}_{t_{11}}(\langle u, 1\rangle.1)]\\
=& s(s(y))[y := s(y)]\\
=& s(s(s(y)))\\
\neq& wert^{\Delta}_{t_{01}}(\langle u, 1\rangle.\varepsilon)\\
wert^{\Delta}_{t_{11}}(\langle \alpha_0, \varepsilon\rangle.\varepsilon) =& \Xi_{\Delta}(\langle u, 1\rangle(\langle \alpha_0, \varepsilon\rangle.1))\\
=& wert^{\Delta}_{t_{11}}(\langle u, 1\rangle.\varepsilon)[y := wert^{\Delta}_{t_{11}}(\langle \alpha_0, \varepsilon\rangle.1)]\\
=& s(s(s(y)))[y := null]\\
=& s(s(s(null)))\\
\neq& wert^{\Delta}_{t_{01}}(\langle \alpha_0, \varepsilon\rangle.\varepsilon)
\end{aligned}$$

Wegen $\hat{\varphi}(s(s(s(null)))) = 3$ ist $(t_{11}, 3) \in \tau_{tv}(M_{10})$.

■

6.3 Graphrepräsentation der Termsegmentdekoration

Tritt in einer Termsegmentgleichung eine Termsegmentinstanz an der Wurzel des Terms auf der rechten Seite der Termsegmentgleichung auf, so wird nach Definition 6.12 eine simultane Termersetzung angestoßen.

Eine graphische Repräsentation der Werte der Termsegmentinstanzen hat in Bezug auf die Ersetzung von Parametern $y \in Y$ gegenüber der Termdarstellung zwei Vorteile:

1. Die Suche nach Teiltermen, deren Wurzeln mit y beschriftet sind, ist im Graphen schneller als der rekursive Abstieg in einem Baum.

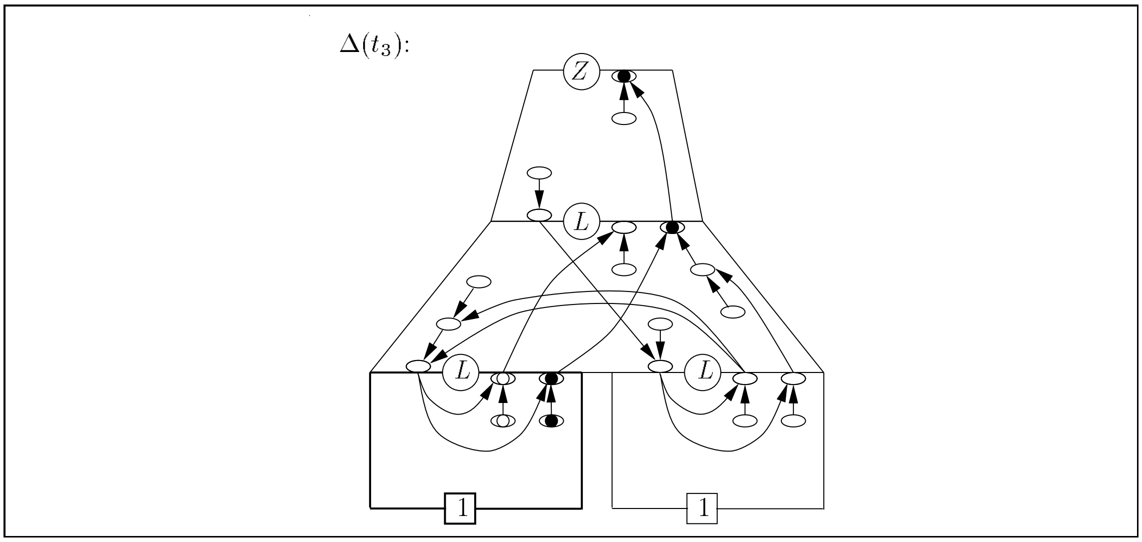


Abbildung 6.3: Voller Abhängigkeitsgraph des Ableitungsbaums t_{11}

2. Erst die graphische Repräsentation ermöglicht es, das mehrfache Vorkommen eines Parameters y in einem Term durch einen einzigen Knoten für y zu repräsentieren, den sich alle Teilterme, in denen y auftritt, gemeinsam teilen (sharing).

Für die Beschreibung der Realisierung der graphischen Repräsentation ist zunächst eine Erweiterung der Terminologie notwendig. Insbesondere ist zu erklären, welche Graphen Terme repräsentieren und wie die Operationen über Bäumen als Operationen über termrepräsentierenden Graphen realisiert werden können.

Intuitiv wird die Graphrepräsentation eines Terms t aus folgenden Teilgraphen konstruiert:

1. Jeder Teilterm t' von t, der nur aus einem Parameter $y \in Y$ besteht, wird durch einen Knoten repräsentiert, der den Wurzelknoten von t' bezeichnet und mit y beschriftet ist.

2. Jeder Teilterm t' der Form $f(t_1, \ldots, t_k)$ mit $k \geq 0$ wird durch einen Graphen repäsentiert, der einen ausgezeichneten Knoten besitzt, der mit f beschriftet ist und der durch genau k Kanten mit den k ausgezeichneten Wurzelknoten der Teilterme t_1 bis t_k verbunden ist.

Diese Darstellung entspricht genau der Repräsentation, die in Definition 2.5 als synonym für die Termdarstellung in Form einer gültigen Zeichenkette über den Symbolen aus Σ und den Parametern aus X angegeben wurde.

Interessant wird erst die Übertragung der simultanen Termersetzung von Definition 2.8 in eine Operation über den termrepräsentierenden Graphen.

Sei (V_0, E_0) der Graph, der den Term t_0 repräsentiert. Sei $var(t_0) = \{y_1, \ldots, y_m\}$, $m \geq 0$, die Menge der Parameter von t_0. Seien $(V_1, E_1), \ldots, (V_m, E_m)$ eine Menge weiterer Graphen, die die Terme $t_1, \ldots, t_m$ repräsentieren.

Um den Term $t_0(t_1, \ldots, t_m)$ graphisch durch (V, E) zu repräsentieren, sind dann intuitiv folgende Schritte nötig: Zuerst muß die Vereinigung aller Graphen gebildet werden:

$$(V, E) = (V_0, E_0) \cup (V_1, E_1) \cup \ldots \cup (V_m, E_m)$$

Der Graph (V, E) enthält dann $m + 1$ „Inseln“, die im zweiten Schritt verbunden werden. Dazu wird jeder Knoten $v \in V_0$, von dem eine Kante ausgeht, die in einem Knoten $v_{y_i} \in V_0$ endet, der mit y_i beschriftet ist, also

$$(v, v_{y_i}) \in E_0 \text{ mit } label(v_{y_i}) = y_i \text{ und } 1 \leq i \leq m$$

mit dem ausgezeichneten Wurzelknoten $v_i \in V_i$ verbunden. Daraus resultiert der Graph (V', E') mit

$$(V', E') = (V, E) \cup (\{v, v_1, \ldots, v_m\}, \{(v, v_i)\})$$

Im dritten und letzten Schritt müssen die überflüssigen Parameterknoten und die Kanten, die in ihnen enden, wieder aus (V', E') entfernt werden. Die überflüssigen Parameterknoten sind genau die Knoten $v_{y_i} \in V_0$, die mit den zu ersetzenden Parametern $\{y_1, \ldots, y_m\}$ beschriftet sind:

$$\begin{aligned} V'' &= V' \setminus \{v_{y_i} \mid v_{y_i} \in V_0, (v, v_{y_i}) \in E_0, label(v_{y_i}) = y_i\} \\ E'' &= E' \lceil V'' \rceil \end{aligned}$$

Bei Verwendung der „graphischen Termersetzung“ zur Konstruktion und inkrementellen Anpassung der Werte der Termsegmentinstanzen ist zusätzlich zu beachten, daß die Modifikation stets nur an *Kopien des Terms* vorgenommen wird, dessen Parameter ersetzt werden sollen. Die Manipulation des Originalgraphen des Terms ist wegen der Elimination der Knoten, die mit Parametern $y \in Y$ beschriftet sind, nur ein einziges Mal möglich.

Die Integration der Graphrepräsentation für die Werte der Termsegmentinstanzen eines Ableitungsbaums in den Algorithmus zur inkrementellen Berechnung der zulässigen Termsegmentdekoration erzeugt einen optimierten inkrementellen Auswerter für geordnete Makroattributgrammatiken.

Literaturverzeichnis

[AV96] I. Adamski and H. Vogler. Macro attribute grammars. Unveröffentlichtes Manuskript, 1996.

[CES88] V. Claus, H. Engesser, and A. Schwill. *Duden Informatik, ein Sachlexikon für Studium und Praxis*. Dudenverlag, 1988.

[Jok95] B. Jokubeit. Speicheroptimierung bei Attributauswertung. Großer Beleg, Fakultät Informatik, Technische Universität Dresden, 1995.

[Kas80] U. Kastens. Ordered attribute grammars. *Acta Informatica*, 13:229–256, 1980.

[KV94] A. Kühnemann and H. Vogler. Synthesized and inherited functions - a new computational modell for syntax-directed semantics. *Acta Informatica*, 31:431–477, 1994.

[KW76] K. Kennedy and S. K. Warren. Automatic generation of efficient evaluators for attribute grammars. *3rd ACM Symposium on Principles of Programming Languages*, pages 32–39, 1 1976.

[RT89] Thomas W. Reps and Tim Teitelbaum. *The Synthesizer Generator - a System for Constructing Language-Based Editors*. Springer, 1989.

[Vog95] H. Vogler. Attributgrammatiken. Vorlesung, persönliche Mitschrift, 1995.

Selbständigkeitserklärung

Hiermit erkläre ich, daß die vorliegende Arbeit von mir selbständig erstellt wurde. Neben den im Literaturverzeichnis angegebenen Quellen wurden keine weiteren von mir zu Hilfe genommen.

Bernd Jokubeit Dresden, 6. Dezember 1996.