Compilerkonstruktion

Wintersemester 2015/16

Prof. Dr. R. Parchmann

13. Oktober 2015

Attributierte Grammatiken

- Erste Ideen zur formalen Definition der Semantik in einer Arbeit von Irons, die Originalarbeit stammt von Knuth aus dem Jahr 1968.

Attributierte Grammatiken

- Erste Ideen zur formalen Definition der Semantik in einer Arbeit von Irons, die Originalarbeit stammt von Knuth aus dem Jahr 1968.
- Idee: Den Symbolen einer Grammatik werden Attribute (Eigenschaften) und den Produktionen Regeln zur Berechnung der Werte dieser Attribute zugeordnet.

Attributierte Grammatiken

- Erste Ideen zur formalen Definition der Semantik in einer Arbeit von Irons, die Originalarbeit stammt von Knuth aus dem Jahr 1968.
- Idee: Den Symbolen einer Grammatik werden Attribute (Eigenschaften) und den Produktionen Regeln zur Berechnung der Werte dieser Attribute zugeordnet.
- Ein Attribut des Startsymbols der Grammatik wird ausgezeichnet. Der Wert dieses Attributs ist dann die Bedeutung des Wortes.

Attributierte Grammatiken

- Erste Ideen zur formalen Definition der Semantik in einer Arbeit von Irons, die Originalarbeit stammt von Knuth aus dem Jahr 1968.
- Idee: Den Symbolen einer Grammatik werden Attribute (Eigenschaften) und den Produktionen Regeln zur Berechnung der Werte dieser Attribute zugeordnet.
- Ein Attribut des Startsymbols der Grammatik wird ausgezeichnet. Der Wert dieses Attributs ist dann die Bedeutung des Wortes.
- Ergebnis: Eine endliche Beschreibung der Bedeutung (Semantik) aller Wörter der von der Grammatik erzeugten Sprache.

Definition

Sei $G = (N, T, P, S)$ eine kontextfreie Grammatik. Jedem Symbol $X \in (N \cup T)$ ist eine endliche Menge von *Attributen* $\mathcal{A}(X)$ zugeordnet.
Das Attribut $a \in \mathcal{A}(X)$ wird auch mit $X.a$ bezeichnet.
Jedem Attribut $a \in \mathcal{A}(X)$ ist eine Wertemenge $\mathcal{W}(X.a)$ zugeordnet.
Die Menge der Attribute $\mathcal{A}(X)$ ist disjunkt zerlegt in

- die Menge der *inheriten Attribute* $\mathcal{A}_I(X)$ und in
- die Menge der *synthetischen Attribute* $\mathcal{A}_S(X)$,

d.h. es gilt $\mathcal{A}(X) = \mathcal{A}_I(X) \cup \mathcal{A}_S(X)$ und $\mathcal{A}_I(X) \cap \mathcal{A}_S(X) = \emptyset$.

Bemerkung

Häufig wird in der Theorie außerdem gefordert, dass das Startsymbol S kein inherites Attribut und jedes $X \in T$ kein synthetisches Attribut besitzt. Es hat sich jedoch als vorteilhaft erwiesen, für praktische Probleme auf diese Einschränkung zu verzichten und statt dessen eine Initialisierung von Atributwerten (etwa durch den Scanner) anzunehmen!

Definition

Jeder Produktion $X \to Y_1 \dots Y_k$ in P, $X \in N$, $Y_i \in (N \cup T)$, $1 \leq i \leq k$, $k \geq 0$, ist eine endliche Menge von Funktionen (Regeln) zugeordnet, die die Berechnung der Werte der synthetischen Attribute von X und der inheriten Attribute aller Y_i festlegen (Semantische Funktionen, semantische Regeln).
Dabei gilt:

- Für jedes Attribut $a \in \mathcal{A}_S(X)$ gibt es eine Regel, die den Wert von a in Abhängigkeit von Werten anderer Attribute $D(X.a) \subseteq \mathcal{A}(X) \cup \bigcup_{i=1}^{k} \mathcal{A}(Y_i)$ berechnet.
- Für jedes Attribut $a \in \mathcal{A}_I(Y_j)$, $1 \leq j \leq k$, gibt es eine Regel, die den Wert von a in Abhängigkeit der Werte anderer Attribute $D(Y_j.a) \subseteq \mathcal{A}(X) \cup \bigcup_{i=1}^{k} \mathcal{A}(Y_i)$ berechnet.

Sprechweise: *Das Attribut a ist abhängig vom Attribut b , falls* $b \in D(X.a)$ *bzw.* $b \in D(Y_i.a)$.

Definition

Eine kontextfreie Grammatik mit einer Attributierung der terminalen und nichtterminalen Symbole und mit einer Zuordnung semantischer Regeln zu allen Produktionen heißt attributierte Grammatik.

Definition

Eine kontextfreie Grammatik mit einer Attributierung der terminalen und nichtterminalen Symbole und mit einer Zuordnung semantischer Regeln zu allen Produktionen heißt attributierte Grammatik.

Definition

Ein attributierter Ableitungsbaum eines Wortes $w \in L(G)$ ist ein Ableitungsbaum für w bzgl. der kontextfreien Grammatik G, bei dem in jedem Knoten für ein $X \in N \cup T$ auch die Attribute $\mathcal{A}(X)$ notiert sind.

Definition

Ein ausgewerteter attributierter Ableitungsbaum eines Wortes $w \in L(G)$ ist ein attributierter Ableitungsbaum für w, in dem jedes Attribut a eines mit $X \in (N \cup T)$ markierten Knotens einen Wert aus $\mathcal{W}(X.a)$ hat und die zugehörige semantische Regel für $X.a$ erfüllt (korrekt) sein muss.

(annotierter Ableitungsbaum)

Der Prozeß des Zuordnens von Werten zu Attributen eines attributierten Ableitungsbaums heißt Auswertung des attributierten Ableitungsbaums.

Bemerkung

Bemerkung

1. *Nicht jeder attributierter Ableitungsbaum läßt sich auswerten!*

Bemerkung

1. *Nicht jeder attributierter Ableitungsbaum läßt sich auswerten!*
2. *Man beachte, dass bei der Verwendung von Funktionen mit Seiteneffekten, z.B. bei der Einführung globaler Variablen, Schwierigkeiten auftreten können und die Auswertung nicht mehr eindeutig ist. In diesem Fall ist die Auswertereihenfolge entscheidend!*

Bemerkung

1. *Nicht jeder attributierter Ableitungsbaum läßt sich auswerten!*
2. *Man beachte, dass bei der Verwendung von Funktionen mit Seiteneffekten, z.B. bei der Einführung globaler Variablen, Schwierigkeiten auftreten können und die Auswertung nicht mehr eindeutig ist. In diesem Fall ist die Auswertereihenfolge entscheidend!*
3. *Häufig wird ein synthetisches Attribut des Startsymbols der Grammatik ausgezeichnet. Dieses Attribut des Wurzelknotens des Ableitungsbaumes enthält dann nach der Auswertung „die Übersetzung“ des Wortes w.*

Beispiel

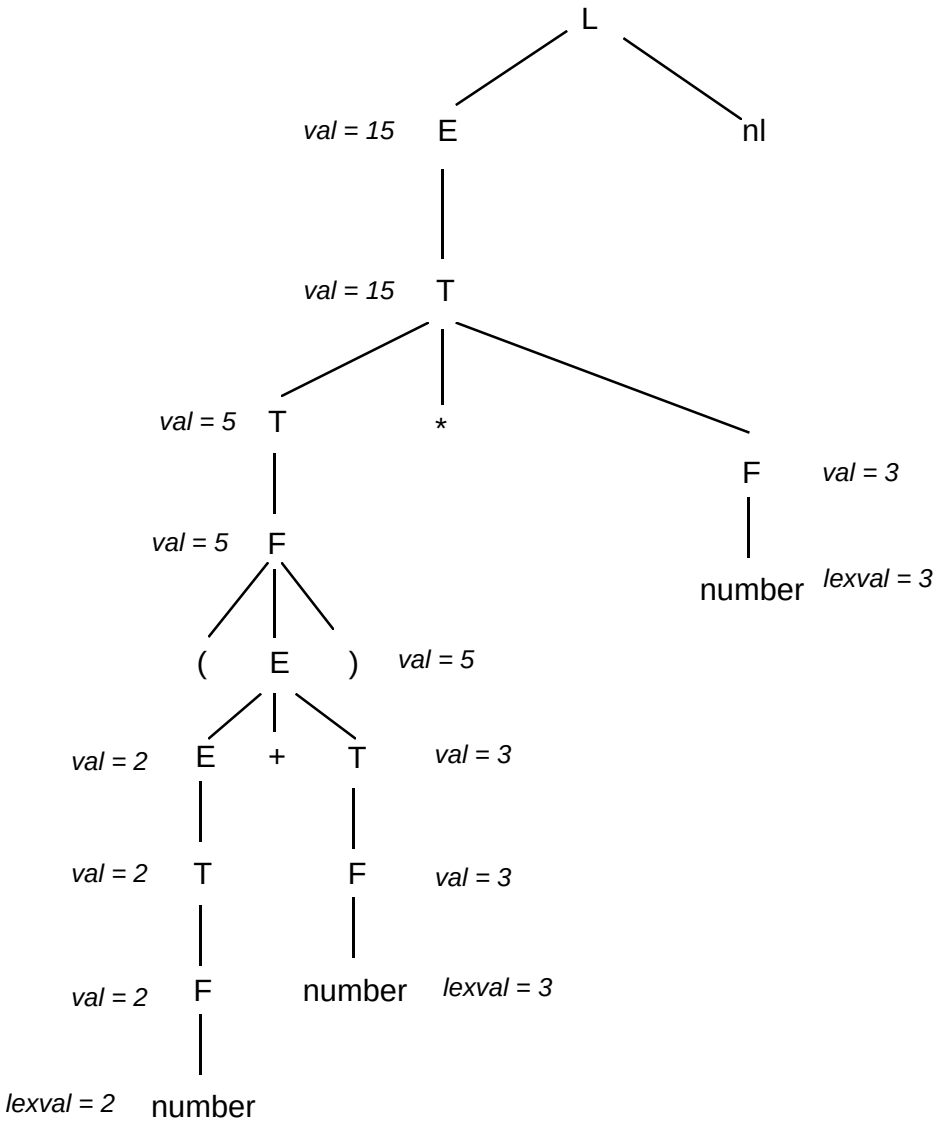
Gegeben ist eine kontextfreie Grammatik $G = (N, T', P, L)$ mit $N = \{L, F, E, T\}$, $T' = \{\textbf{nl}, +, *, (,), \textbf{number}\}$.

Die Attributmengen sind $\mathcal{A}_S(E) = \mathcal{A}_S(F) = \mathcal{A}_S(T) = \{val\}$, $\mathcal{A}_S(\textbf{number}) = lexval$ und
$\mathcal{A}_S(L) = \mathcal{A}_I(L) = \mathcal{A}_I(E) = \mathcal{A}_I(F) = \mathcal{A}_I(T) = \mathcal{A}_I(\textbf{number}) = \emptyset$.

Die folgende Attributierung bewirkt, dass ein Wort aus $L(G)$ als arithmetischer Ausdruck interpretiert und ausgewertet wird. Die der Produktion $L \to E$ **nl** zugeordnete semantische Regel hat als Seiteneffekt, das Ausdrucken des Wertes des arithmetischen Ausdrucks.

Produktionen	Semantische Regeln
$L \to E$ **nl**	print($E.val$)
$E \to E_1 + T$	$E.val := E_1.val + T.val$
$E \to T$	$E.val := T.val$
$T \to T_1 * F$	$T.val := T_1.val * F.val$
$T \to F$	$T.val := F.val$
$F \to (E)$	$F.val := E.val$
$F \to$ **number**	$F.val :=$ **number**.$lexval$

Ausgewerteter attributierter Ableitungsbaum



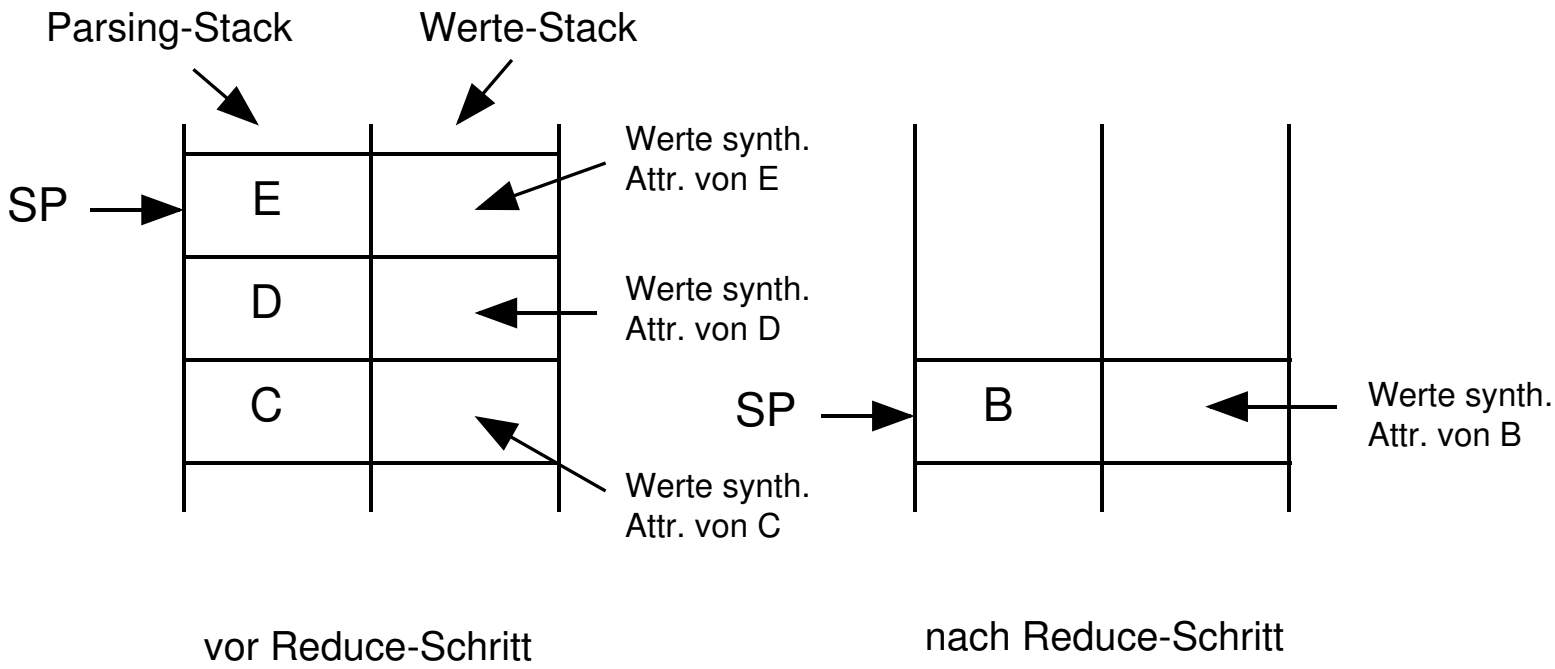
$$w = (2 + 3) * 3 \ \mathbf{nl}.$$

Definition

Eine attributierte Grammatik, in der nur synthetische Attribute definiert sind, heißt S-attributierte Grammatik.

Beispiel

Sei $B \to CDE$ eine Produktion und seien für alle $a \in \mathcal{A}_S(B)$ Regeln zugeordnet.



Beispiel

Vorgehen bei unserer Beispielgrammatik:

- $val(\mathrm{SP})$ bezeichne den obersten Wert im Werte-Stack
- Der lexikale Scanner legt bei einer Zahl den Tokenwert direkt in den Werte-Stack ab.
- SP' bezeichne die Position von SP nach einem Reduce-Schritt.

Dann kann man die semantischen Regeln durch kurze Code-Fragmente angeben, die die semantischen Regeln realisieren.

Darstellung der Regeln durch Code-Fragmente:

Produktionen	Code-Fragmente
$L \to E$ **nl**	print(val[SP − 1])
$E \to E_1 + T$	val[SP′] := val[SP − 2] + val[SP]
$E \to T$	
$T \to T_1 * F$	val[SP′] := val[SP − 2] ∗ val[SP]
$T \to F$	
$F \to (E)$	val[SP′] := val[SP − 1]
$F \to$ **number**	

Abarbeitung des Wortes $(2 + 3) * 3$ **nl**:

Eingabe	Pars.-Stack	Werte-Stack	angew. Prod
$(2 + 3) * 3$ **nl**			
$2 + 3) * 3$ **nl**	$($	$-$	
$+3) * 3$ **nl**	$($**number**	-2	
$+3) * 3$ **nl**	$(F$	-2	$F \rightarrow$ **number**
$+3) * 3$ **nl**	$(T$	-2	$T \rightarrow F$
$+3) * 3$ **nl**	$(E$	-2	$E \rightarrow T$
$3) * 3$ **nl**	$(E+$	$-2-$	
$) * 3$ **nl**	$(E +$ **number**	$-2 - 3$	
$) * 3$ **nl**	$(E + F$	$-2 - 3$	$F \rightarrow$ **number**
$) * 3$ **nl**	$(E + T$	$-2 - 3$	$T \rightarrow F$
$) * 3$ **nl**	$(E$	-5	$E \rightarrow E_1 + T$
$*3$ **nl**	(E)	$-5-$	
$*3$ **nl**	F	5	$F \rightarrow (E)$
$*3$ **nl**	T	5	$T \rightarrow F$
3 **nl**	$T*$	$5-$	

usw.

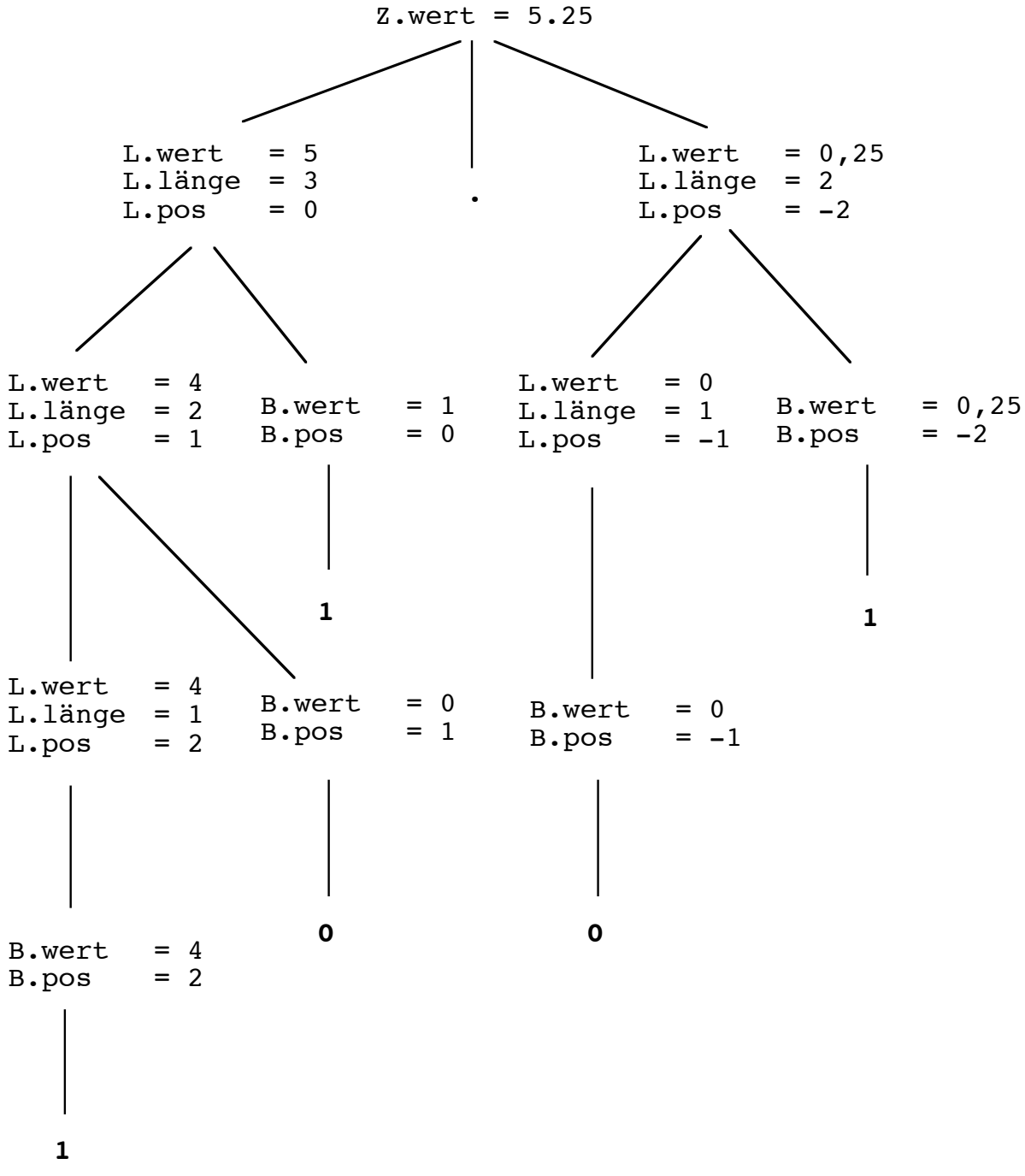
Beispiel

Sei $G = (N, T, P, Z)$ eine kontextfreie Grammatik mit $N = \{Z, L, B\}$, $T = \{0, 1, .\}$ und

$$\begin{array}{ll} \mathcal{A}_S(L) = \{\mathit{länge}, \mathit{wert}\} & \mathcal{A}_I(L) = \{\mathit{pos}\} \\ \mathcal{A}_S(Z) = \{\mathit{wert}\} & \mathcal{A}_I(Z) = \emptyset \\ \mathcal{A}_S(B) = \{\mathit{wert}\} & \mathcal{A}_I(B) = \{\mathit{pos}\} \end{array}$$

Diese Grammatik erzeugt Wörter, die als Darstellung von Festkommazahlen im Dualsystem interpretiert werden können. Dabei steht das Symbol Z für eine Festkommazahl, L für eine Liste von 0 und 1 und B für ein Bit, also für 0 oder 1. Als Übersetzung eines Wortes soll der Wert der dargestellten Zahl berechnet und im synthetischen Attribut *wert* von Z abgelegt werden.

Produktion	**Semantische Regeln**
$Z \to L_1 \; . \; L_2$	$Z.wert := L_1.wert + L_2.wert$ $L_1.pos := 0$ $L_2.pos := -L_2.länge$
$L \to L_1 B$	$L.länge := L_1länge + 1$ $L.wert := L_1.wert + B.wert$ $L_1.pos := L.pos + 1$ $B.pos := L.pos$
$L \to B$	$L.länge := 1$ $L.wert := B.wert$ $B.pos := L.pos$
$B \to 0$	$B.wert := 0$
$B \to 1$	$B.wert := 2^{B.pos}$


Z.wert = 5.25
L.wert = 5
L.länge = 3
L.pos = 0
.
L.wert = 0,25
L.länge = 2
L.pos = -2
L.wert = 4
L.länge = 2
L.pos = 1
B.wert = 1
B.pos = 0
L.wert = 0
L.länge = 1
L.pos = -1
B.wert = 0,25
B.pos = -2
1
1
L.wert = 4
L.länge = 1
L.pos = 2
B.wert = 0
B.pos = 1
B.wert = 0
B.pos = -1
0
0
B.wert = 4
B.pos = 2
1

Abhängigkeitsgraphen

Der Abhängigkeitsgraph $g(\pi)$ für die Produktion π wird wie folgt bestimmt:

```
for jedes Symbol X in π  do
    for jedes Attribut a von X do
        erzeuge einen neuen Knoten in g(π)
        mit Markierung X.a
for jede Regel ρ zu π do
    bestimmt ρ das Attribut X.a in Abhängigkeit
    der Attribute X_1.b_1, ... , X_k.b_k ,
    so erzeuge eine Kante in g(π) von jedem der
    Knoten X_i.b_i nach X.a
```

Der Abhängigkeitsgraph $G(\tau)$ für den Ableitungsbaum τ wird wie folgt bestimmt:

Erzeuge zunächst die Knoten $S.a$ für jedes Attribut a des Startsymbols S.
Dann durchlaufe τ in Präordnung
    Sei n der nächste Knoten in Präordnung mit Markierung X in τ.
    Ist X ein nichtterminales Symbol, dann existieren in γ die Knoten $X.b$, b Attribut von X.
    Wurde auf X in τ die Produktion π angewendet, so füge $g(\pi)$ zu $G(\tau)$ hinzu, wobei die Knoten $X.b$ in $G(\tau)$ und $g(\pi)$ identifiziert werden

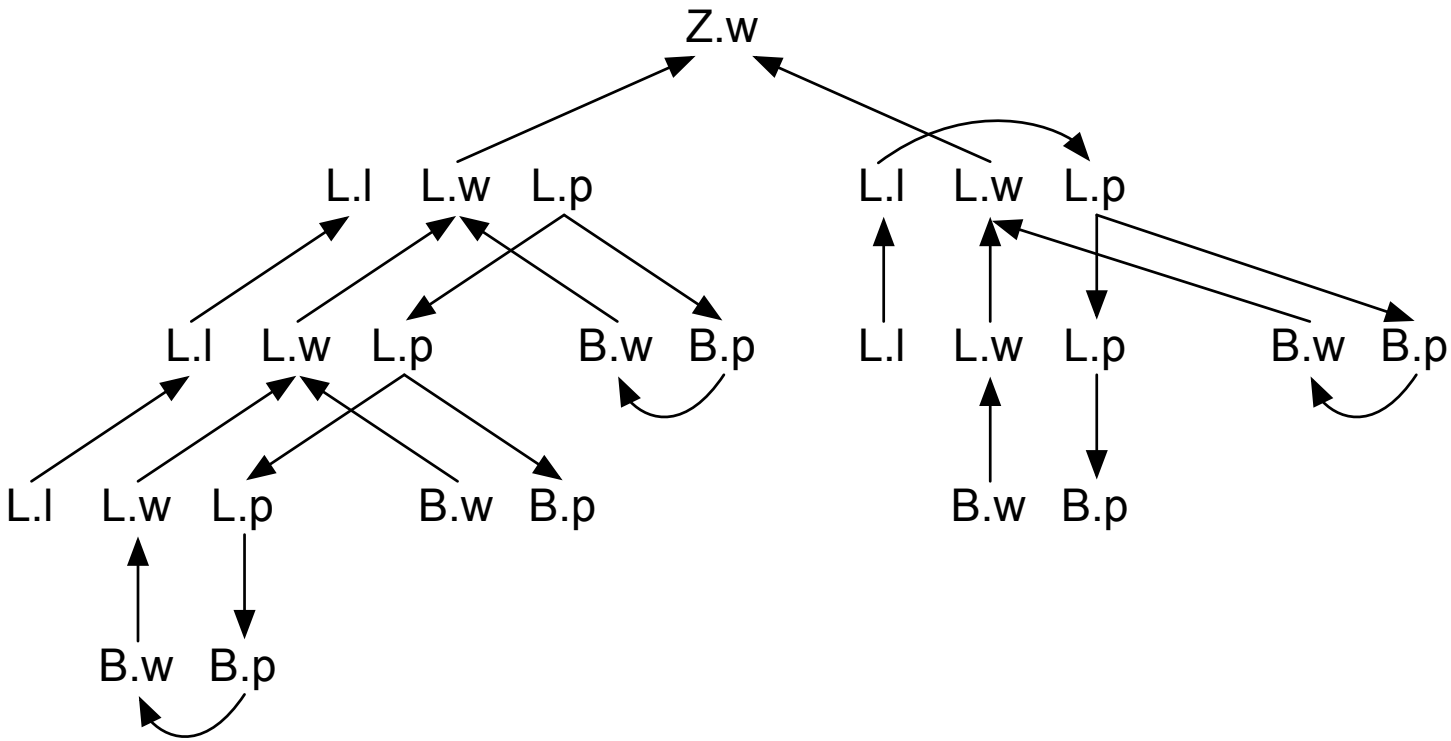
Der Abhängigkeitsgraph eines Ableitungsbaumes kann nun benutzt werden, um eine Auswertereihenfolge für die Attribute im attributierten Ableitungsbaum festzulegen.

Es gilt:
Gibt es eine Kante von $X.a$ nach $Y.b$ im Abhängigkeitsgraphen, so muss das Attribut a von X vor dem Attribut b von Y mit einem Wert versehen werden.

Damit folgt sofort:

- Es existiert genau dann eine Auswertereihenfolge, wenn der Abhängigkeitsgraph des Ableitungsbaums azyklisch ist.

Abhängigkeitsgraph für w = 101.01



Übersetzung eines Wortes w mit einer attributierten Grammatik

Übersetzung eines Wortes w mit einer attributierten Grammatik

1. Parsen des vorgelegten Wortes w bzgl. der kontextfreien Grammatik und Konstruktion des Ableitungsbaums

Übersetzung eines Wortes w mit einer attributierten Grammatik

1. Parsen des vorgelegten Wortes w bzgl. der kontextfreien Grammatik und Konstruktion des Ableitungsbaums
2. Konstruktion des Abhängigkeitsgraphen und Festlegung einer Auswertereihenfolge

Übersetzung eines Wortes w mit einer attributierten Grammatik

1. Parsen des vorgelegten Wortes w bzgl. der kontextfreien Grammatik und Konstruktion des Ableitungsbaums
2. Konstruktion des Abhängigkeitsgraphen und Festlegung einer Auswertereihenfolge
3. Auswertung der Attribute ergibt die Übersetzung als Wert eines ausgezeichneten synthetischen Attributs des Startsymbols.

Methoden der Implementation einer derartigen Übersetzung

Methoden der Implementation einer derartigen Übersetzung

1. *on-the-fly Methoden*, bei denen während des Parsings alle Attribute berechnet werden können.

Methoden der Implementation einer derartigen Übersetzung

1. *on-the-fly Methoden*, bei denen während des Parsings alle Attribute berechnet werden können.
2. *tree-walk Methoden*, bei denen in einem oder mehreren Durchläufen durch den Ableitungsbaum die Attribute ausgewertet werden

Methoden der Implementation einer derartigen Übersetzung

1. *on-the-fly Methoden*, bei denen während des Parsings alle Attribute berechnet werden können.
2. *tree-walk Methoden*, bei denen in einem oder mehreren Durchläufen durch den Ableitungsbaum die Attribute ausgewertet werden
   - die Anzahl der Durchläufe ist unbegrenzt.

Methoden der Implementation einer derartigen Übersetzung

1. *on-the-fly Methoden*, bei denen während des Parsings alle Attribute berechnet werden können.
2. *tree-walk Methoden*, bei denen in einem oder mehreren Durchläufen durch den Ableitungsbaum die Attribute ausgewertet werden
   - die Anzahl der Durchläufe ist unbegrenzt.
   - die Anzahl der benötigten Durchläufe ist durch eine Konstante beschränkt

Weiterhin kann man noch bzgl. der Reihenfolge differenzieren, in der die Nachfolgerknoten eines Knotens im Ableitungsbaum ausgewertet werden, z.B.

- immer links-nach-rechts

Weiterhin kann man noch bzgl. der Reihenfolge differenzieren, in der die Nachfolgerknoten eines Knotens im Ableitungsbaum ausgewertet werden, z.B.

- immer links-nach-rechts
- immer rechts-nach-links

Weiterhin kann man noch bzgl. der Reihenfolge differenzieren, in der die Nachfolgerknoten eines Knotens im Ableitungsbaum ausgewertet werden, z.B.

- immer links-nach-rechts
- immer rechts-nach-links
- alternierend bei jedem Durchlauf

Weiterhin kann man noch bzgl. der Reihenfolge differenzieren, in der die Nachfolgerknoten eines Knotens im Ableitungsbaum ausgewertet werden, z.B.

- immer links-nach-rechts
- immer rechts-nach-links
- alternierend bei jedem Durchlauf
- in einer durch eine Permutation für jede Produktion festgelegten Reihenfolge.

Weiterhin kann man noch bzgl. der Reihenfolge differenzieren, in der die Nachfolgerknoten eines Knotens im Ableitungsbaum ausgewertet werden, z.B.

- immer links-nach-rechts
- immer rechts-nach-links
- alternierend bei jedem Durchlauf
- in einer durch eine Permutation für jede Produktion festgelegten Reihenfolge.

Weiterhin kann man unterscheiden, ob diese Reihenfolge statisch, also für alle Ableitungsbäume der Grammatik, oder dynamisch, je nach vorliegenden Ableitungsbaum, gewählt wird.

L-attributierte Grammatiken

Definition

Eine attributierte Grammatik $G = (N, T, P, S)$ heißt L-attributiert, wenn für jede Produktion $X \to Y_1 \dots Y_k$ in P mit $X \in N$ und $Y_i \in (N \cup T)$, $1 \leq i \leq k$, $k \geq 0$ und für jedes j, $1 \leq j \leq k$ gilt:

Ist $a \in \mathcal{A}_I(Y_j)$, dann ist a nur abhängig von Attributen aus $\mathcal{A}_I(X) \cup \bigcup_{i=1}^{j-1} \mathcal{A}(Y_i)$

L-attributierte Grammatiken

Definition

Eine attributierte Grammatik $G = (N, T, P, S)$ heißt L-attributiert, wenn für jede Produktion $X \to Y_1 \dots Y_k$ in P mit $X \in N$ und $Y_i \in (N \cup T)$, $1 \leq i \leq k$, $k \geq 0$ und für jedes j, $1 \leq j \leq k$ gilt:

Ist $a \in \mathcal{A}_I(Y_j)$, dann ist a nur abhängig von Attributen aus $\mathcal{A}_I(X) \cup \bigcup_{i=1}^{j-1} \mathcal{A}(Y_i)$

Bemerkung

Jede S-attributierte Grammatik ist auch L-attributiert.

Definition

Ein syntax-gesteuertes Übersetzungsssschema (SDTS) ist andere Darstellung einer attributierten Grammatik $G = (N, T, P, S)$.
Die kontextfreien Produktionen $X \to Y_1 \dots Y_k$, $X \in N$, $Y_i \in N \cup T$, $1 \leq i \leq k$, $k > 0$ werden um die semantischen Regeln erweitert. Die Produktionen haben dann die Form $X \to \alpha_0 Y_1 \alpha_1 \dots \alpha_{k-1} Y_k \alpha_k$, wobei die α_i die Form $\alpha_i = \epsilon$ oder $\alpha_i = \{$ semantische Regeln $\}$ haben.
Die α_i heißen Aktionen.

Bedeutung:

Führe die semantischen Regeln in α_i aus, nachdem die Ableitung für Y_i erstellt wurde und bevor die Ableitung von Y_{i+1} beginnt.

Offensichtlich erfüllt für ein korrektes SDTS die folgenden Regeln:
Für alle $1 \leq i \leq k$ muß gelte:

Offensichtlich erfüllt für ein korrektes SDTS die folgenden Regeln: Für alle $1 \leq i \leq k$ muß gelte:

- der Wert eines inherites Attributs a von Y_i muß in einer der Aktionen $\alpha_0, \ldots \alpha_{i-1}$ berechnet werden, üblicherweise in α_{i-1}, und

Offensichtlich erfüllt für ein korrektes SDTS die folgenden Regeln: Für alle $1 \leq i \leq k$ muß gelte:

- der Wert eines inherites Attributs a von Y_i muß in einer der Aktionen $\alpha_0, \ldots \alpha_{i-1}$ berechnet werden, üblicherweise in α_{i-1}, und
- der Wert eines synthetischen Attributs b kann erst berechnet werden, wenn alle zur Berechnung benötigten Werte bekannt sind, üblicherweise also in α_k.

Offensichtlich erfüllt für ein korrektes SDTS die folgenden Regeln: Für alle $1 \leq i \leq k$ muß gelte:

- der Wert eines inherites Attributs a von Y_i muß in einer der Aktionen $\alpha_0, \ldots \alpha_{i-1}$ berechnet werden, üblicherweise in α_{i-1}, und
- der Wert eines synthetischen Attributs b kann erst berechnet werden, wenn alle zur Berechnung benötigten Werte bekannt sind, üblicherweise also in α_k.

Bemerkung

Jede L-attributierte Grammatik lässt sich in ein SDTS umschreiben.

Auswertung L-attributierter Grammatiken

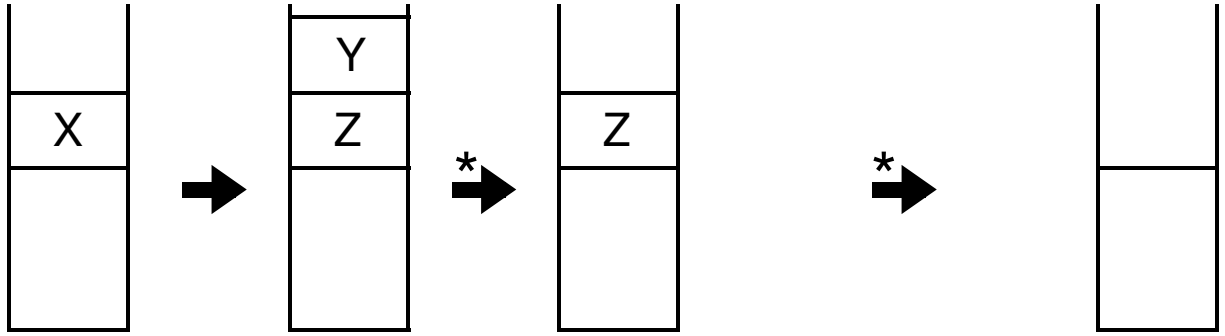
Wenn man eine LL-Grammatik mit einer L-Attributierung gegeben hat, kann man den Parsing- und den Auswerteprozess parallel durchführen. Man benötigt in diesem Fall einen zusätzlichen Attributwerte-Stack.

Beispiel

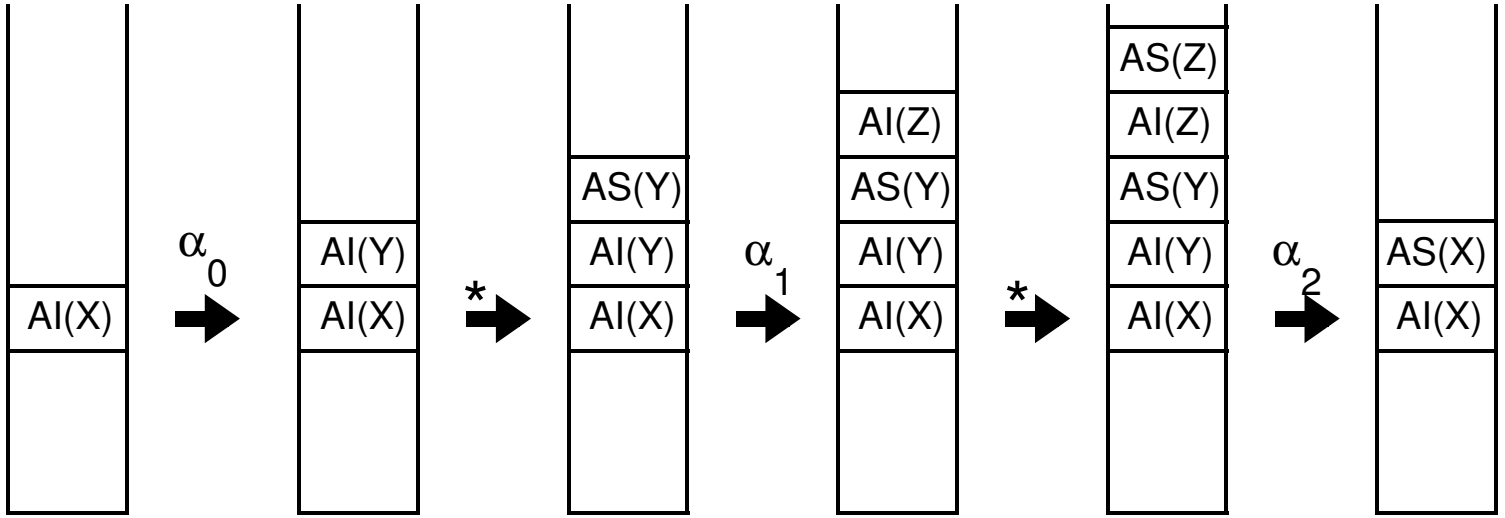
Sei $X \to \alpha_0 Y \alpha_1 Z \alpha_2$ eine Produktion einer L-Attributierten Grammatik.
Anfangssituation: Auf dem Parsing-Stack steht X und auf dem Werte-Stack steht ein Record mit den Werten der inheriten Attribute von X.

Angewendete Produktion $X \to \alpha_0 Y \alpha_1 Z \alpha_2$

Parsing-Stack:



Werte-Stack:



Einfacher und natürlicher ist die Verbindung einer L-Attributierung mit einem Parser, der nach der Methode des rekursiven Abstiegs arbeitet.

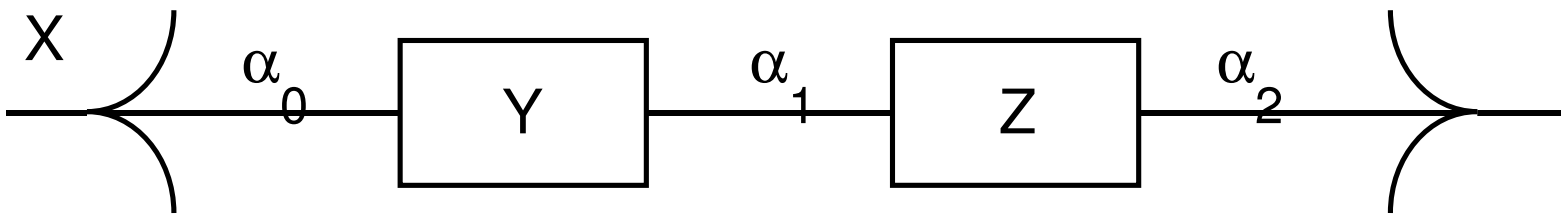
Als Beispielproduktion sei $X \to \alpha_0 Y \alpha_1 Z \alpha_2$ gewählt, wobei $X, Y, Z \in N$ gelte und $\alpha_0, \alpha_1, \alpha_2$ semantische Aktionen sind.

Einfacher und natürlicher ist die Verbindung einer L-Attributierung mit einem Parser, der nach der Methode des rekursiven Abstiegs arbeitet.

Als Beispielproduktion sei $X \to \alpha_0 Y \alpha_1 Z \alpha_2$ gewählt, wobei $X, Y, Z \in N$ gelte und $\alpha_0, \alpha_1, \alpha_2$ semantische Aktionen sind.

Wie üblich wird zunächst für jedes nichtterminale Symbol ein Syntaxgraph konstruiert. Man markiert dabei zusätzlich die Kanten des Graphen mit den Aktionen.

Ausschnitt aus dem Syntaxgraph für X:



Für jeden Syntaxgraphen eines nichtterminales Symbols X erzeugt man dann wie üblich eine Prozedur mit Namen `X`.

- Diese Prozedur bekommt für jedes inherite Attribut in $\mathcal{A}_I(X)$ einen Formalparameter und gibt die Werte der synthetischen Attribute in $\mathcal{A}_S(X)$ (z.B. als Record) zurück.
- Weiterhin hat die Prozedur `X` lokale Variablen für jedes synthetische Attribut von X und jeweils eine lokale Variable für jedes Attribut eines jeden im Syntaxgraphen auftretenden Symbols. (Tritt ein Symbol mehrfach im Syntaxgraphen auf, so müssen entsprechend viele Exemplare der lokalen Variablen erzeugt werden.)

Neben dem üblichen Code zur Steuerung der Ableitung beim rekursiven Abstieg werden

- bei terminalen Symbolen a die Werte der synthetischen Attribute von a in die lokalen Variablen gespeichert,
- bei nichtterminalen Symbolen Y bekommen Prozeduraufrufe die Form $\texttt{syn}_1\texttt{, ..., syn}_r\texttt{ := Y(inh}_1\texttt{, ...,inh}_k\texttt{)}$, ($\texttt{inh}_1\texttt{, ..., inh}_k$ sind die lokalen Variablen für die inheriten und $\texttt{syn}_1\texttt{, ..., syn}_r$ für die synthetischen Attribute von Y)
- die Aktionen durch entsprechende Code-Fragmente in die Parsing-Prozedur übertragen
- beim Verlassen der Prozedur die Werte der synthetischen Attribute zurückgegeben

Umformungen attributierter Grammatiken

Hier sollen exemplarisch nur zwei Fälle betrachtet werden:

1. Das Entfernen von Aktionen, die in der rechten Seite von Produktionen eines bottom-up parsebaren SDTS auftreten, um ein deterministisches Bottom-Up-Parsing mit gleichzeitiger Attributauswertung zu ermöglichen.

Umformungen attributierter Grammatiken

Hier sollen exemplarisch nur zwei Fälle betrachtet werden:

1. Das Entfernen von Aktionen, die in der rechten Seite von Produktionen eines bottom-up parsebaren SDTS auftreten, um ein deterministisches Bottom-Up-Parsing mit gleichzeitiger Attributauswertung zu ermöglichen.
2. Das Entfernen linksrekursiver Produktionen bei S-attributierten Grammatiken, damit ein deterministisches Top-Down-Parsing möglich wird.

Entfernen von Aktionen aus der Mitte von Produktionen

- Man ersetzt eine Aktion a, die nicht am Ende, sondern innerhalb einer Produktion ausgeführt werden muss, durch ein neues nichtterminales Symbol (Marker) und führt eine ε-Produktion für dieses Symbol mit Aktion a ein.

Entfernen von Aktionen aus der Mitte von Produktionen

- Man ersetzt eine Aktion a, die nicht am Ende, sondern innerhalb einer Produktion ausgeführt werden muss, durch ein neues nichtterminales Symbol (Marker) und führt eine ε-Produktion für dieses Symbol mit Aktion a ein.
- Die zusätzlich eingeführten Symbole und Produktionen erlauben weiterhin ein deterministisches bottom-up Parsen, wobei die semantichen Aktionen jetzt wie üblich bei jedem Reduktionsschritt ausgeführt werden.

Beispiel

Die folgenden Regeln eines SDTS

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +T\{\text{print}('+')\}R \mid -T\{\text{print}('-')\}R \mid \varepsilon \\
T &\rightarrow \text{number}\ \{\text{print}(\text{number.val}\}
\end{aligned}
$$

werden transformiert zu

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +TMR \mid -TNR \mid \varepsilon \\
T &\rightarrow \text{number}\ \{\text{print}(\text{number.val}\} \\
M &\rightarrow \varepsilon\ \{\text{print}('+')\} \\
N &\rightarrow \varepsilon\ \{\text{print}('-')\}
\end{aligned}
$$

Entfernen linksrekursiver Produktionen

Seien $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m$ alle A-Produktionen, deren rechte Seite mit A beginnt und seinen $A \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ alle restlischen A-Produktionen. Weiterhin gelte $\alpha_i \neq \varepsilon$ für $1 \leq i \leq m$. Man ersetzt diese Produktionen durch

$$A \to \beta_1 A', \ \ldots, \ A \to \beta_n A' \quad \text{und}$$
$$A' \to \alpha_1 A', \ \ldots, \ A' \to \alpha_m A', \ A' \to \varepsilon.$$

wobei A' ein neues nichtterminales Symbol ist.
Ist die ursprüngliche linksrekursive Grammatik S-attributiert, so kann man durch Umformungen der Regeln die neue Grammatik L-attributiert machen.

Beispiel

Die auftretenden nichtterminalen Symbole haben wie das Token **number** alle nur das synthetische Attribut *val*.

Produktion	sem. Regel
$E \to E_1 + T$	$E.val := E_1.val + T.val$
$E \to E_1 - T$	$E.val := E_1.val - T.val$
$E \to T$	$E.val := T.val$
$T \to (E)$	$T.val := E.val$
$T \to \mathbf{number}$	$T.val := \mathbf{number}.val$

Entfernt man die Links-Rekursionen und ändert die Attributierung entsprechend, so erhält man eine rechts-rekursive Grammatik. Das neu hinzugekommene nichtterminale Symbol R bekommt ein inherites Attribut i und ein synthetisches Attribut s. Insgesamt erhät man eine äquivalente L-attributierte Grammatik, die in Form eines SDTS notiert wird.

$$
\begin{aligned}
E &\rightarrow T\ \{R.i := T.val\}\ R\ \{E.val := R.s\} \\
R &\rightarrow +T\ \{R_1.i := R.i + T.val\}\ R_1\ \{R.s := R_1.s\} \\
R &\rightarrow -T\ \{R_1.i := R.i - T.val\}\ R_1\ \{R.s := R_1.s\} \\
R &\rightarrow \epsilon\ \{R.s := R.i\} \\
T &\rightarrow (E\ \{T.val := E.val\}\ ) \\
T &\rightarrow \textbf{number}\ \{T.val := \textbf{number}.val\}
\end{aligned}
$$