Semantische Analyse – Teil 2

Übersicht

- Attributgrammatik
- Eigenschaften von Attributgrammatiken
- Semantik von Attributgrammatiken
- Beispiele von Attributgrammatiken
- Generierung von Attributauswertern
- Attributabhängigkeiten
- Attributauswertung
- Unterklassen von Attributgrammatiken
  - L-attributierte Grammatiken
  - LL-attributierte Grammatiken
  - LR-attributierte Grammatiken
  - Attributierung in YACC/Bison

Lernziele

- Das Grundprinzip und die wichtigsten theoretischen Aspekte von Attribut-grammatiken beherrschen
- Grundideen und wichtigste Algorithmen zur Bestimmung von Attributabhängig-keiten und zur Attributauswertung benennen können
- Das Prinzip der L-Attributierung und seine Integration in die Syntax-Analyse wiedergeben können

Attributgrammatik (1/3)

Attributgrammatik intuitiv

- „Erweitert“ kfG um „semantische Information“
- Assoziiert „Attribute“ als Träger (statischer) semantischer Information mit den Symbolen der zugrunde liegenden kfG
- Gibt funktionale Abhängigkeiten zwischen den Werten von Attributvorkommen in den Produktionen der Grammatik an
  (mit Bedingungen für die Auswertbarkeit aller Attributexemplare)

„Fluss der Information durch die Produktionen“

Beachte Begriffsbildungen

Präzise Definitionen, s.u.

- *Attribut* ☛ : allgemeiner Begriff
- *Attributvorkommen* ☛ : Vorkommen eines Attributs in einer Produktion
- *Attributexemplar* ☛ : Vorkommen eines Attributs im Syntaxbaum

Attributgrammatik (2/3)

Attributgrammatik formal (Teil 1)

- Sei $G = (V_N, V_T, P, S)$ eine kfG mit $V =_{def} V_N \cup V_T$
- Für jedes Symbol $X \in V$
  - Inh(X): Menge der **ererbten** (*inherited*) **Attribute** von X
  - Syn(X): Menge der **abgeleiteten** (*synthesized*) **Attribute** von X mit $Inh(X) \cap Syn(X) = \varnothing$
  - $Attr(X) =_{def} Inh(X) \cup Syn(X)$: **Attribute** von X
- Für kfG G
  - $Inh =_{def} \bigcup_{X \in V} Inh(X)$, $Syn =_{def} \bigcup_{X \in V} Syn(X)$, $Attr =_{def} Inh \cup Syn$
- Für jedes Attribut $a \in Attr$
  - D_a: **Wertebereich** für a

Beispiel

- Geg.: kfG G_0 mit Produktionen
  - $E \rightarrow E + T$, $E \rightarrow T$, $p{:}T \rightarrow T * F$, $q{:}T \rightarrow F$, $F \rightarrow (E)$, $F \rightarrow$ **id**
- Attribute
  - $Syn(X) =_{def} \{val \mid X \in V_N \cup \{\textbf{id}\}\}$
  - $Inh(X) =_{def} \{depth \mid X \in V_N \cup \{\textbf{id}\}\}$
- Wertebereiche von val, depth: nat

Attributgrammatik (3/3)

Attributgrammatik formal (Teil 2)

- Für jede Produktion p: $X_0 \rightarrow X_1 \ldots X_{n_p}$, mit $X_i \in V$, $1 \le i \le n_p$, $X_0 \in V_N$
  - a_i: **Vorkommen** des Attributs a in p (falls $a \in Attr(X_i)$)
  - V(p): Menge aller Attributvorkommen in p
  - **Semantische Regeln** $a_i = f_{p,a,i}(b^1_{j_1}, \ldots, b^k_{j_k})$ $(0 \le j_l \le n_p)$ $(1 \le l \le k)$
    für jedes Vorkommen a_i $(0 \le i \le n_p)$, wobei
    - $a \in Inh(X_i)$ (für $1 \le i \le n_p$)
    - $a \in Syn(X_0)$ (für $i = 0$)
    - $f_{p,a,i}: D_{b1} \times \ldots \times D_{bk} \rightarrow D_a$

$\in Attr(X_{j_l})$

D.h. definiert werden nur
- abgeleitete Attributvorkommen für die linke Seite
- ererbte Attributvorkommen für die rechte Seite

Beispiel (Fortsetzung)

- Geg.: kfG G_0 mit $E \rightarrow E + T$, $E \rightarrow T$, p:$T \rightarrow T * F$, q:$T \rightarrow F$, $F \rightarrow (E)$, $F \rightarrow$ **id**
- Attributvorkommen
  - $V(p) = \{val_0, val_1, val_3\} \cup \{depth_0, depth_1, depth_3\}$
  - $V(q) = \{val_0, val_1\} \cup \{depth_0, depth_1\}$
- Semantische Regeln
  - Für p: $val_0 = val_1 \times val_3$
    $depth_1 = depth_0 + 1$
    $depth_3 = depth_0 + 1$
  - Für q: $val_0 = val_1$
    $depth_1 = depth_0 + 1$

Attributgrammatik (3/3)

Beispiel

- Geg.: kfG G_0 mit Produktionen
  - $E \rightarrow E + T$, $E \rightarrow T$, $p{:}T \rightarrow T * F$, $q{:}T \rightarrow F$, $F \rightarrow (E)$, $F \rightarrow \mathbf{id}$
- Attribute
  - $Syn(X) =_{def} \{val \mid X \in V_N \cup \{\mathbf{id}\}\}$
  - $Inh(X) =_{def} \{depth \mid X \in V_N \cup \{\mathbf{id}\}\}$
- Wertebereiche von val, depth: nat
- Attributvorkommen
  - $V(p) = \{val_0, val_1, val_3\} \cup \{depth_0, depth_1, depth_3\}$
  - $V(q) = \{val_0, val_1\} \cup \{depth_0, depth_1\}$
- Semantische Regeln
  - Für p: $val_0 = val_1 \times val_3$
    $depth_1 = depth_0 + 1$
    $depth_3 = depth_0 + 1$
  - Für q: $val_0 = val_1$
    $depth_1 = depth_0 + 1$

Attributgrammatik (3/3)

Beispiel

- Geg.: kfG G_0 mit Produktionen
  - $E \rightarrow E + T$, $E \rightarrow T$, $p{:}T \rightarrow T * F$, $q{:}T \rightarrow F$, $F \rightarrow (E)$, $F \rightarrow \mathbf{id}$
- Attribute
  - $Syn(X) =_{def} \{val \mid X \in V_N \cup \{\mathbf{id}\}\}$
  - $Inh(X) =_{def} \{depth \mid X \in V_N \cup \{\mathbf{id}\}\}$
- Wertebereiche von val, depth: nat
- Attributvorkommen
  - $V(p) = \{val_0, val_1, val_3\} \cup \{depth_0, depth_1, depth_3\}$
  - $V(q) = \{val_0, val_1\} \cup \{depth_0, depth_1\}$
- Semantische Regeln
  - Für p: $val_0 = val_1 \times val_3$
    $depth_1 = depth_0 + 1$
    $depth_3 = depth_0 + 1$
  - Für q: $val_0 = val_1$
    $depth_1 = depth_0 + 1$

Eigenschaften von Attributgrammatiken (1/2)

Definierende und angewandte Attributvorkommen, Normalform

- Geg.: Attributgrammatik AG, Produktion p: $X_0 \rightarrow X_1 ... X_{n_p}$
- Dann
  - **Definierende Attributvorkommen**: a_i mit $a \in Inh(X_i)$ $(1 \leq i \leq n_p)$ und a_0 mit $a \in Syn(X_0)$
  - **Angewandte Attributvorkommen**: alle anderen Vorkommen
  - AG **ist in Normalform**: alle Argumente semantischer Regeln sind angewandte Vorkommen

D.h. z.B. die Attributvorkommen, die durch semantische Regeln definiert werden

Intuitiv

- Attributgrammatik ist in Normalform $\Leftrightarrow$
  für jedes definierende Attributvorkommen in einer Produktion gibt es genau eine semantische Regel und deren Argumente sind ausschließlich angewandte Vorkommen

Beispiel

- Geg.: Produktion p:T $\rightarrow$ T * F mit semantischen Regeln

  $val_0 = val_1 \times val_3$
  $depth_1 = depth_0 + 1$
  $depth_3 = depth_0 + 1$
- Dann
  - Definierend: val_0, $depth_1$, $depth_3$
  - Angewandt: val_1, val_3, $depth_0$

Damit Normalform-Eigenschaft für p *erfüllt*

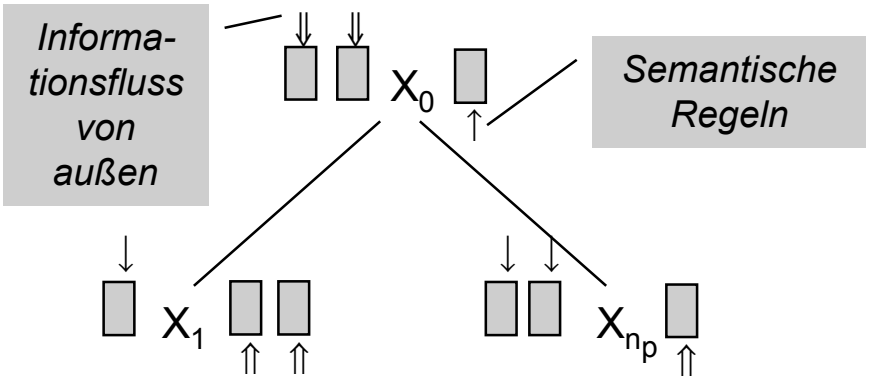
Eigenschaften von Attributgrammatiken (2/2)

Abgeleitete Attribute bei Terminalen

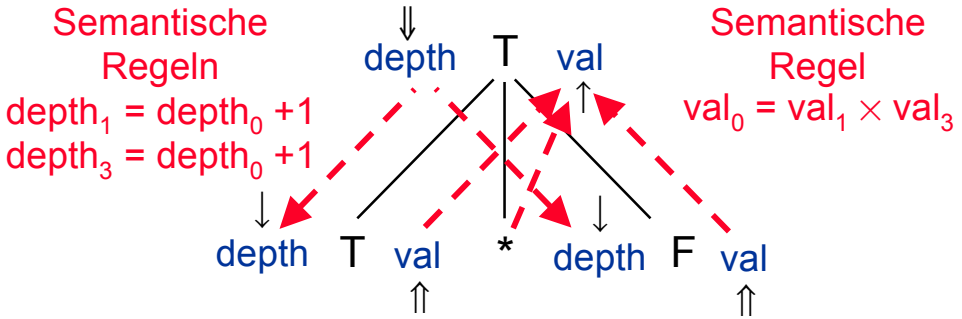
- Beispiele
  - Wert von Konstanten, Externdarstellung von Namen, Adressen von Stringkonstanten, ...
- Kommen nur als Argumente semantischer Regeln vor (werden nicht durch Regeln definiert)
- Wertzuordnung durch externe Regeln (in der Praxis: vom Scanner geliefert)

Graphische Darstellung attributierter Produktionen

- Produktion als Baum dargestellt
- Ererbte Attributvorkommen: links (von den Knoten im Baum)
- Abgeleitete Attributvorkommen: rechts (von den Knoten im Baum)



Schematischer Zusammenhang



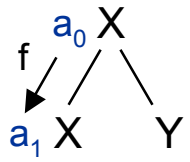
Beispiel

Semantik von Attributgrammatiken (1/3)

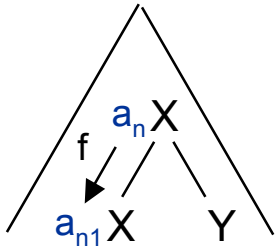
Begriffe

- **Attribut** zu Symbol X
  Element aus Attr(X)
- **Attributvorkommen** a_i von a zu X in Produktion p
  Eindeutige Zuordnung von a zu Vorkommen i von X in p
- **Attributexemplar**
  Vorkommen a_{ni} von a in zu Knoten n gehöriger Produktion p im Syntaxbaum t
- Beispiel
  - $a \in Inh(X)$
  - p: $X \rightarrow XY$
  - $a_1 = f(a_0)$: semantische Regel zu p

Attribut



Attributvorkommen



Attributexemplar

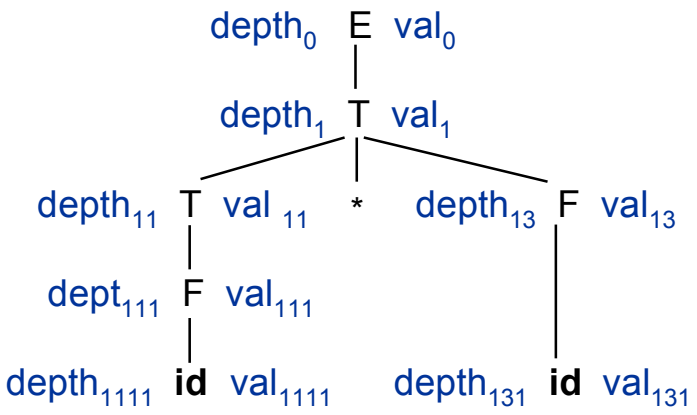
Semantik von Attributgrammatiken (2/3)

Basis der Semantik einer Attributgrammatik (formal)

- Funktion value: Attributexemplare → Wertebereich
  - Für Syntaxbaum t definiert durch Gleichungssystem über allen Attributexemplaren in t als Unbekannten
  - Gleichungen $value(a_{ni}) = f_{p,a,i}(value(b^1_{nj_1}), ..., value(b^k_{nj_k}))$ für Knoten n und zu n gehöriger Produktion p sind durch semantische Regeln $a_i = f_{p,a,i}(b^1_{j_1}, ..., b^k_{j_k})$ induziert

Beispiel

Attributierter Baum



Gleichungssystem

Für val

$val_0 = val_1$
$val_1 = val_{11} \times val_{13}$
$val_{11} = val_{111}$
$val_{111} = val_{1111}$ $\quad val_{1111} = ...$
$val_{13} = val_{131}$ $\quad val_{131} = ...$

Analog für depth

Semantik von Attributgrammatiken (3/3)

Eigenschaften des Gleichungssystems

- Zyklisch (oder rekursiv): keine oder mehrere Lösungen
- Azyklisch (oder nicht-rekursiv): genau eine Lösung (genau ein Wert pro Attributexemplar)

Wohlgeformt

- Attributgrammatik heißt **wohlgeformt**, wenn für **alle** Syntaxbäume t das zugehörige Gleichungssystem azyklisch ist

Semantik (einer wohlgeformten Attributgrammatik)

- Eindeutige Zuordnung von Attributwerten zu den Knoten der Syntaxbäume (der zugrunde liegenden kfG)

Beispiele von Attributgrammatiken (1/4)

Beispiele von Attributgrammatiken

- Varianten des Typberechnungsproblems für arithmetische Ausdrücke
- Konversion von Binärbrüchen in Dezimalzahlen

Dabei

- Definition der funktionalen Beziehungen zwischen Attributvorkommen durch funktionale Sprache mit üblichen Eigenschaften, d.h.
  - Fallunterscheidung durch „pattern matching“
  - Unwesentliche Argumente durch „_“ bezeichnet
  - Bei Überlappung verschiedener Fälle, Auswahl „speziellerer“ Fälle vor „allgemeineren“
  - Mehrfach vorkommende Variablen müssen beim matching auf die gleichen Werte treffen

Beispiele von Attributgrammatiken (2/4)

Attribute grammar AG_1;
Nonterminals E, T, F, P, Aop, Mop;
Attributes **syn** typ **with** E, T, F, P, c, v **domain** {int, real}
syn op **with** Aop, Mop **domain** {'+', '–', '*', '/', 'div'}

Rules

1:	E → E Aop T	E_0.typ = f_1(E_1.typ, T.typ)
2:	E → T	E.typ = T.typ
3:	T → T Mop F	T_0.typ = f_3(Mop.op, T_1.typ, F.typ)
4:	T → F	T.typ = F.typ
5:	F → P	F.typ = P.typ
6:	P → c	P.typ = c.typ
7:	P → v	P.typ = v.typ
8:	P → (E)	P.typ = E.typ
9:	Aop → +	Aop.op = '+'
10:	Aop → –	Aop.op = '–'
11:	Mop → *	Mop.op = '*'
12:	Mop → /	Mop.op = '/'
13:	Mop → **div**	Mop.op = 'div'

Functions

f_1 int int = int	f_3 '*' int int = int
f_1 _ _ = real	f_3 'div' int int = int
	f_3 '*' _ _ = real
	f_3 '/' _ _ = real

Beobachtungen

- Hier nur abgeleitete Attribute
- Großteil der semantischen Regeln sind Gleichungen (identische Übergabe von Werten)

Konvention (zur Schreibabkürzung)

- Fehlen einer semantischen Regel für (definierendes) Attributvorkommen bedeutet identische Übergabe von gleichnamigem Vorkommen:
  - Ererbt links → ererbt rechts
  - Abgeleitet rechts → abgeleitet links

Beispiele von Attributgrammatiken (3/4)

Attribute grammar AG_3;
Nonterminals S, E, T, F, P, Aop, Mop;
Attributes **syn** typ **with** E, T, F, P, c, v **domain** {int, real}
syn op **with** Aop, Mop **domain** {'+', '–', '*', '/', 'div'}
inh obltyp **with** E, T, F, P **domain** {int, unspec}

Rules

0:	S → E	$E.obltyp = unspec$			
1:	E → E Aop T	$E_0.typ = f_1(E_1.typ, T.typ)$			
2:	E → T				
3':	T → T Mop F	$T_0.typ = f_{31}(Mop.op, T_0.obltyp, T_1.typ, F.typ)$, $T_1.obltyp = f_{32}(Mop.op, T_0.obltyp)$, $F.obltyp = f_{32}(Mop.op, T_0.obltyp)$			
4:	T → F				
5:	F → P				
6:	P → c	$P.typ = f_6(P.obltyp, c.typ)$			
7:	P → v	$P.typ = f_6(P.obltyp, v.typ)$			
8:	P → (E)				
9:	Aop → +	Aop.op = '+'	10:	Aop → –	Aop.op = '–'
11:	Mop → *	Mop.op = '*'	12:	Mop → /	Mop.op = '/'
13:	Mop → **div**	Mop.op = 'div'			

Functions

f_1 int int = int	f_{31} '*' int int = int	f_{31} 'div' _ int int = int	f_{31} '*' _ _ _ = real	f_{31} '/' unspec _ _ = real	f_{32} 'div' _ = int	f_{32} _ int = int	f_{32} _ _ = unspec	f_6 int int = int	f_6 unspec t = t
f_1 _ _ = real									

Beispiele von Attributgrammatiken (4/4)

Attributgrammatik

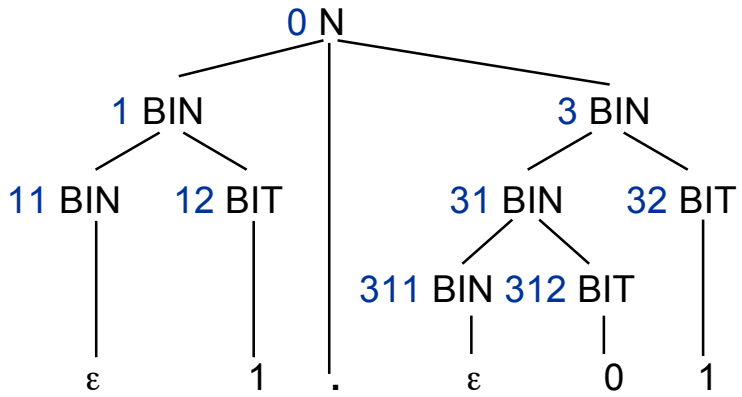
Attribute grammar Bin_to_Dec;
Nonterminals N, BIN, BIT;
Attributes **syn** l **with** BIN **domain** {int}
syn v **with** N, BIN, BIT **domain** {real}
inh r **with** BIN, BIT **domain** {int}

Rules

1:	N → BIN.BIN	$N.v = BIN_1.v + BIN_3.v$
		$BIN_1.r = 0$
		$BIN_3.r = -BIN_3.l$
2:	BIN → BIN BIT	$BIN_0.v = BIN_1.v + BIT.v$
		$BIN_0.l = BIN_1.l + 1$
		$BIN_1.r = BIN_0.r + 1$
		$BIT.r = BIN_0.r$
3:	BIN → ε	$BIN.v = 0$
		$BIN.l = 0$
4:	BIT → 1	$BIT.v = 2^{BIT.r}$
5:	BIT → 0	$BIT.v = 0$

Intuition:
- v = „value“ (Dezimalwert)
- r = „rank“ (Stellenwert)
- l = „length“ (Länge)

Syntaxbaum (Beispiel)



Gleichungssystem (22 Gleichungen mit 22 Unbekannten)

0 $v_0 = \cancel{v_1 + v_3}$ 5/41
$r_1 = 0$
$r_3 = -\cancel{l_3}$ 2

$v_1 = \cancel{v_{11} + v_{12}}$ 1 3
$l_1 = \cancel{l_{11} + 1}$ 1
$r_{11} = \cancel{r_1 + 1}$ 1
$r_{12} = \cancel{r_1}$ 0

$v_3 = \cancel{v_{31} + v_{32}}$ 1/4
$l_3 = \cancel{l_{31} + 1}$ 2
$r_{31} = \cancel{r_3 + 1}$ –1
$r_{32} = \cancel{r_3}$ –2

11 $v_{11} = 0$
$l_{11} = 0$

12 $v_{12} = \cancel{2^{r_{12}}}$ 1

31 $v_{31} = \cancel{v_{311} + v_{312}}$ 0
$l_{31} = \cancel{l_{311} + 1}$ 1
$r_{311} = \cancel{r_{31} + 1}$ 0
$r_{312} = \cancel{r_{31}}$ –1

311 $v_{311} = 0$
$l_{311} = 0$

32 $v_{32} = \cancel{2^{r_{32}}}$ 1/4

312 $v_{312} = 0$

Generierung von Attributauswertern

Aufgabe

- Auswertung von Attributexemplaren in Syntaxbäumen t
  - Attributgrammatik definiert Gleichungssystem für t mit Attributexemplaren als Unbekannten
  - Damit: Auswertung von Attributexemplaren = Lösung dieses Gleichungssystems

Problem: Wie löst man ein solches Gleichungssystem?

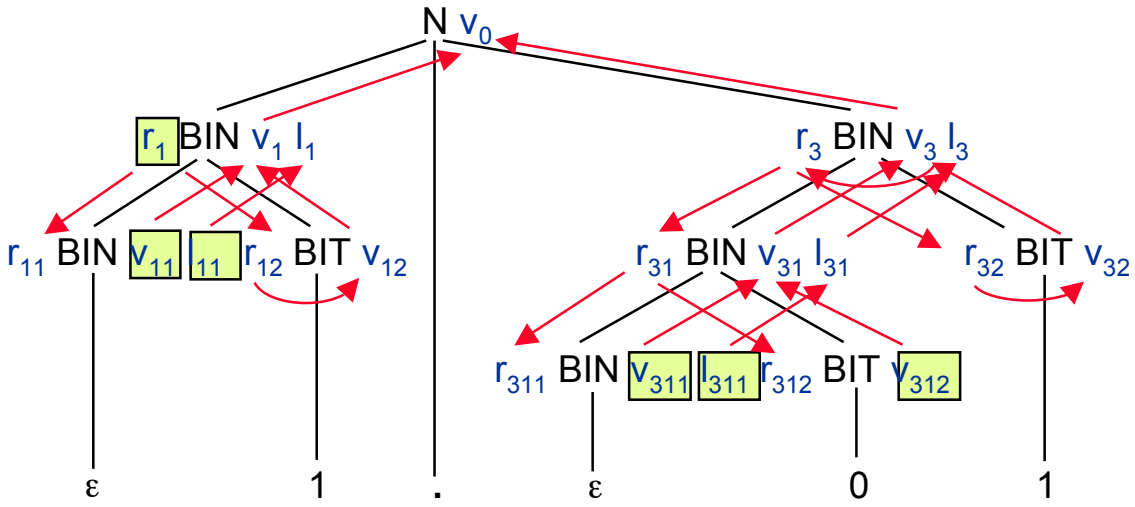
- Annahme: Gleichungssystem azyklisch

Lösungsmöglichkeiten

- Dynamisch: Eliminationsverfahren — *Wie im letzten Beispiel*
  - Problem (in jedem Schritt):
    Suchen der nächsten zu eliminierenden Unbekannten
- Statisch
  - Analyse der *Abhängigkeiten zwischen Attributexemplaren* ☛ aufgrund der semantischen Regeln
  - Statisch bestimmte „Besuchsreihenfolge“
  - Attributauswerter, der diese Reihenfolge benutzt

Abhängigkeiten zwischen Attributexemplaren

Beispiel (Abhängigkeiten zwischen Attributexemplaren)



Attribute grammar Bin_to_Dec;
Nonterminals N, BIN, BIT;
Attributes **syn** l **with** BIN **domain** {int}
syn v **with** N, BIN, BIT
domain {real}
inh r **with** BIN, BIT **domain** {int}

Rules

1:	N → BIN.BIN	$N.v = BIN_1.v + BIN_3.v$
		$BIN_1.r = 0$
		$BIN_3.r = -BIN_3.l$
2:	BIN → BIN BIT	$BIN_0.v = BIN_1.v + BIT.v$
		$BIN_0.l = BIN_1.l + 1$
		$BIN_1.r = BIN_0.r + 1$
		$BIT.r = BIN_0.r$
3:	BIN → ε	$BIN.v = 0$
		$BIN.l = 0$
4:	BIT → 1	$BIT.v = 2^{BIT.r}$
5:	BIT → 0	$BIT.v = 0$

Unterschiedliche Abhängigkeiten

- Lokal: innerhalb von Produktionen
- Global: durch Anwendungskontext
  (Teilbaum oder oberes Baumfragment)

Attributabhängigkeiten (1/7)

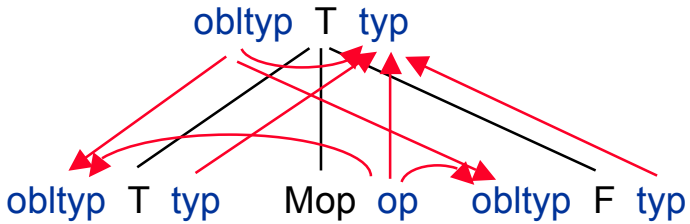
Produktionslokale Abhängigkeit

- **Produktionslokale Abhängigkeitsrelation** $Dp(p) \subseteq V(p) \times V(p)$
  - Definiert durch $b_j\ Dp(p)\ a_i \Leftrightarrow a_i = f_{p,a,i}(..., b_j, ...)$
  - Dargestellt durch produktionslokalen Abhängigkeitsgraphen

Menge aller Attributvorkommen in p

Beispiel (AG_3)

- Geg.: Produktion 3‘: $T \rightarrow T\ Mop\ F$ mit
  - $T_0.typ = f_{31}(Mop.op, T_0.obltyp, T_1.typ, F.typ)$
  - $T_1.obltyp = f_{32}(Mop.op, T_0.obltyp)$
  - $F.obltyp = f_{32}(Mop.op, T_0.obltyp)$
- Dp(3‘)



Im folgenden vorausgesetzt: Attributgrammatiken in Normalform

- D.h. Argumente in den semantischen Regeln immer angewandte Vorkommen
  - ⇒ Dp(p) nichtreflexiv (für alle p)
  - ⇒ Graph ist nicht zyklisch und alle Wege haben Länge 1

Attributabhängigkeiten (2/7)

Beispiel (Attributgrammatik Bin_to_Dec)

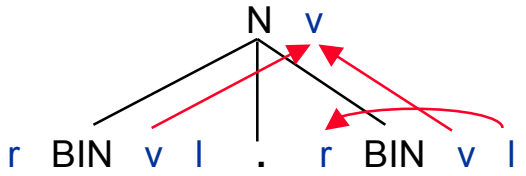
Attribute grammar Bin_to_Dec;
Nonterminals N, BIN, BIT;
Attributes **syn** l **with** BIN **domain** {int}
syn v **with** N, BIN, BIT **domain** {real}
inh r **with** BIN, BIT **domain** {int}

Rules

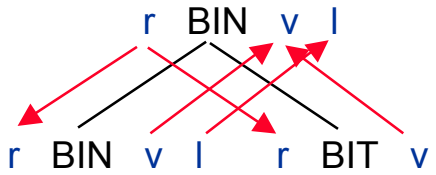
1:	$N \rightarrow BIN.BIN$	$N.v = BIN_1.v + BIN_3.v$
		$BIN_1.r = 0$
		$BIN_3.r = -BIN_3.l$
2:	$BIN \rightarrow BIN\ BIT$	$BIN_0.v = BIN_1.v + BIT.v$
		$BIN_0.l = BIN_1.l + 1$
		$BIN_1.r = BIN_0.r + 1$
		$BIT.r = BIN_0.r$
3:	$BIN \rightarrow \varepsilon$	$BIN.v = 0$
		$BIN.l = 0$
4:	$BIT \rightarrow 1$	$BIT.v = 2^{BIT.r}$
5:	$BIT \rightarrow 0$	$BIT.v = 0$

Produktionslokale Abhängigkeitsgraphen Dp(i)

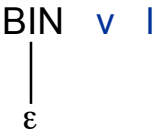
Dp(1)



Dp(2)

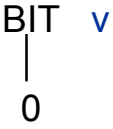


Dp(3)



Dp(4)

Dp(5)



Es gilt

- Produktionslokale Abhängigkeiten induzieren Abhängigkeiten zwischen Attributexemplaren in Syntaxbäumen

Attributabhängigkeiten (3/7)

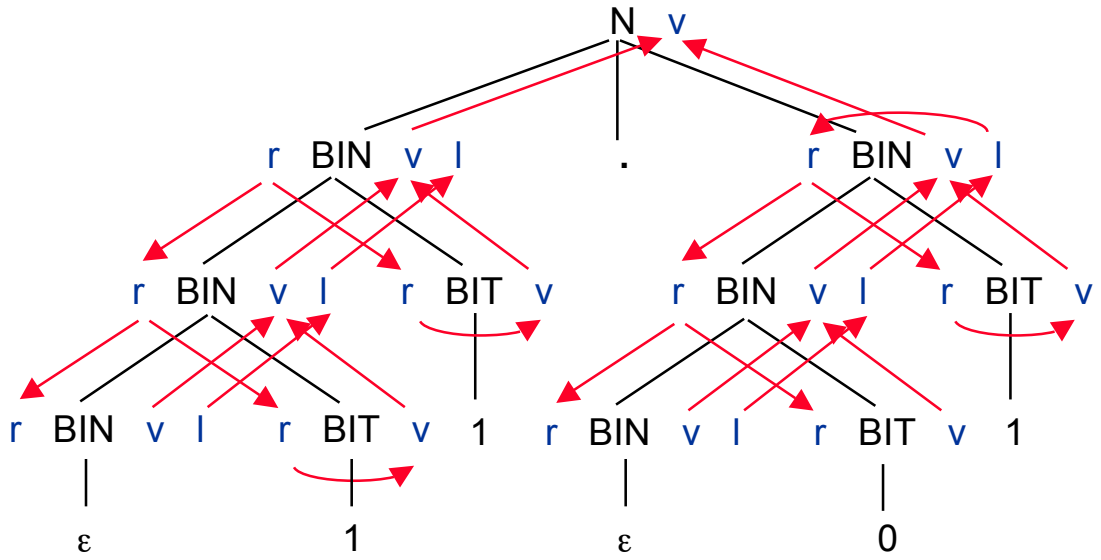
Ziel

- Attributwerte für Syntaxbäume berechnen

Individueller Abhängigkeitsgraph

- Geg.: Attributgrammatik AG mit zugrunde liegender kfG G und Syntaxbaum t gemäß G
- **Individueller Abhängigkeitsgraph** Dt(t) ergibt sich durch „Zusammenkleben“ der produktionslokalen Abhängigkeitsgraphen der angewandten Produktionen

Beispiel (Attributgrammatik Bin_to_Dec)



Attributabhängigkeiten (4/7)

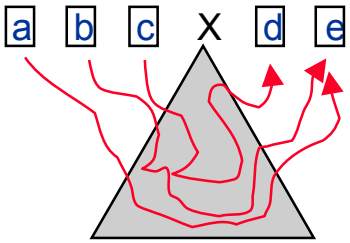
Zyklenfrei

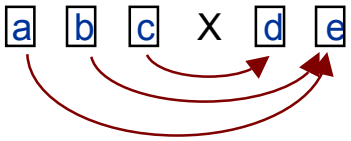
Individueller Abhängigkeitsgraph

- Eine Attributgrammatik ist **zyklenfrei**, wenn Dt(t) für jeden Baum t der zugrunde liegenden kfG das zugehörige Gleichungssystem azyklisch ist
- D.h.: „wohlgeformt“ und „zyklenfrei“ sind Synonyme

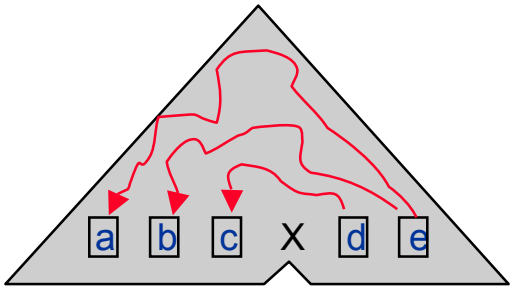
Abhängigkeiten von Attributexemplaren

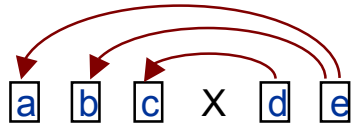
- Abgeleitete von den ererbten (in einem Teilbaum): *unterer charakteristischer Graph* ☛
- Ererbte von den abgeleiteten (in einem Baumfragment): *oberer charakteristischer Graph* ☛





unterer charakte-ristischer Graph





oberer charakte-ristischer Graph

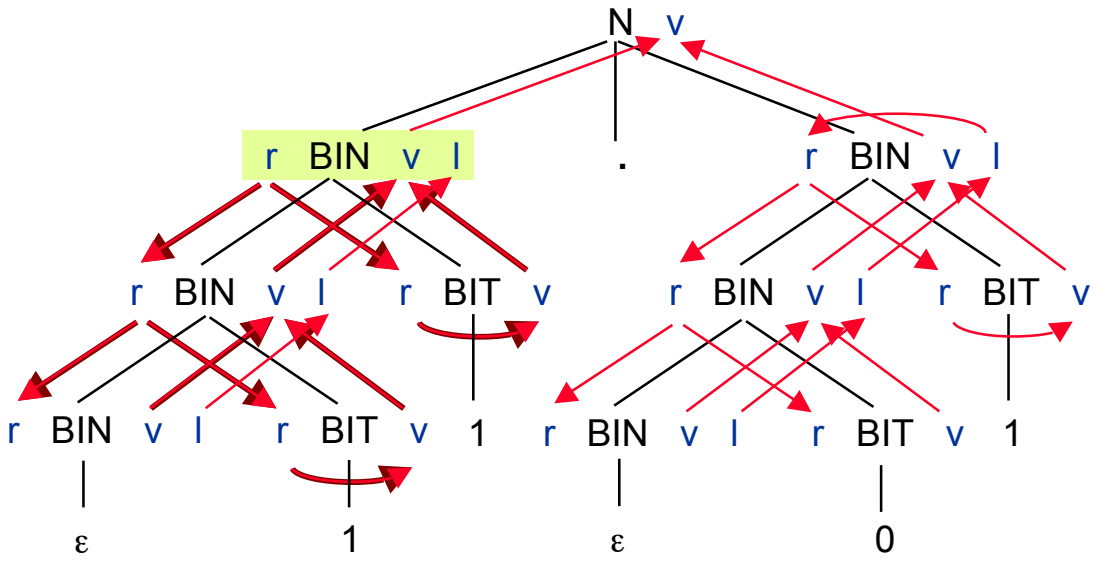
Attributabhängigkeiten (5/7)

Unterer charakteristischer Graph (für Nichtterminal X)

- Geg.: Baum t der zugrunde liegenden kfG mit Wurzelmarkierung X
- Von t induzierter **unterer charakteristischer Graph**
  - Graph der Relation $Dt\uparrow_t(X) \subseteq Inh(X) \times Syn(X)$
  - $Dt\uparrow_t(X) =_{def}$ Einschränkung der transitiven Hülle von Dt(t) auf die Attributexemplare von X
- Effiziente Vorausberechnung (zur Generierungszeit) durch „Grammatikflussanalyse“

Indirekter Zusammenhang
ererbt → abgeleitet

Beispiel (Attributgrammatik Bin_to_Dec)



Unterer charakteristischer Graph für BIN

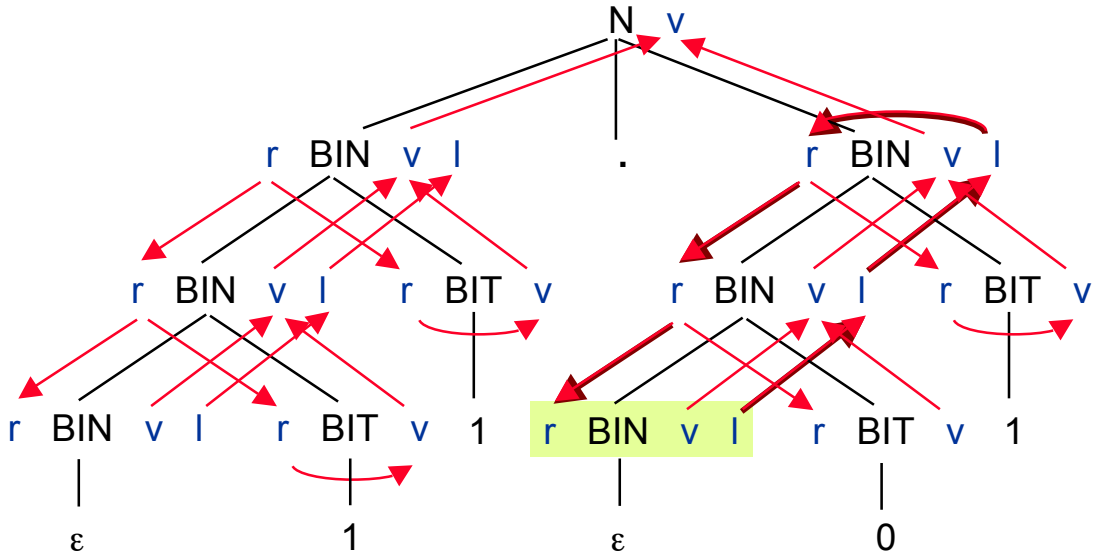
Attributabhängigkeiten (6/7)

Oberer charakteristischer Graph (für Nichtterminal X)

- Geg.: Baum t/n der zugrunde liegenden kfG mit Markierung X am Knoten n, bei dem der an n hängende Teilbaum entfernt wurde („oberes Baumfragment“)
- Von t induzierter **oberer charakteristischer Graph**
  - Graph der Relation $Dt\downarrow_{t,\,n}(X) \subseteq Syn(X) \times Inh(X)$
  - $Dt\downarrow_{t,\,n}(X) =_{def}$ Einschränkung der transitiven Hülle von Dt(t/n) auf die Attributexemplare von n
- Effiziente Vorausberechnung (zur Generierungszeit) durch „Grammatikflussanalyse“

Indirekter Zusammenhang abgeleitet → ererbt

Beispiel (Attributgrammatik Bin_to_Dec)



Oberer charakteristischer Graph für BIN

Attributabhängigkeiten (7/7)

Abhängigkeiten zwischen Attributvorkommen

- Geg.: Knoten n (mit Produktion p) in Baum t
- **Abhängigkeitsrelation** EDp(p) für p bei Vorkommen an n = Kombination von
  - Produktionslokaler Abhängigkeitsgraph Dp(p)
  - Oberer charakteristischer Graph $Dt\downarrow_{t,\,n}$ an n
  - Untere charakteristische Graphen $Dt\uparrow_{t}(X)$ an den Kindern von n

Es gilt

- Eine AG ist zyklenfrei ⇔
  alle Graphen EDp(p) sind nichtzyklisch (für alle Produktionen p)
- Transitive Hülle jeder Abhängigkeitsrelation zu einer zyklenfreien Attributgrammatik ist eine partielle Ordnung — Folgt aus Azyklizität
- Diese partielle Ordnung kann zu einer totalen Ordnung gemacht werden (s.u.)

Z.B. durch „Topologisches Sortieren“

Auswertungsordnung

Auswertungsordnung

Legt Reihenfolge der Attributauswertung fest

- **Auswertungsordnung einer Relation** $R \subseteq V \times V =_{def}$ totale Ordnung T auf V mit $R \subseteq T$
- **Auswertungsordnung für Baum t** $=_{def}$
  totale Ordnung T(t) auf der Menge V(t) aller Attributexemplare von t mit $Dt(t) \subseteq T(t)$

Individueller Abhängigkeitsgraph auf den Attributexemplaren von t

Beachte

- Auswertungsordnung vom Baum t abhängig
  (da untere / obere charakteristische Graphen von t abhängig)
- Insgesamt aber nur endlich viele Ordnungen
  (da Zahl der Attribute zu jedem Symbol X endlich und fest)

Allerdings „zu viele“, nämlich 2^{nm} *für jedes Symbol* X
(*wobei* n = |Inh(X)| *und* m = |Syn(X)|)

Alternative: Approximative globale Abhängigkeiten

- „billigere“ Alternativen zu charakteristischen Graphen mit weniger oder mehr Kanten
- Probleme
  - Kanten weglassen: vergebliche Besuche von Unterbäumen
  - Kanten hinzufügen: evtl. Zyklen im Abhängigkeitsgraph (und damit ewiges Warten des Attributauswerters)

Attributauswertung (1/4)

Aufgabe der Attributauswertung

- Geg.: attributierter Baum = Syntaxbaum (der zugrunde liegenden kfG) erweitert um Speicherzellen für Attributexemplare
  (an jedem Knoten n: Verbund mit Komponenten für die Attribute von symb(n))
- Ges.: Werte für Attributexemplare berechnen und (an den Knoten) abspeichern

Grundvoraussetzung

- Jeder Baum muss „Auswertungsordnung“ besitzen ⇔ AG ist zyklenfrei

Attributauswertung (2/4)

(konzeptionelle) **Phasen der Attributauswertung** (in einem Syntaxbaum t)

Können praktisch auch „verschmolzen“ sein

- *Strategiephase*: Bestimmung einer Auswertungsordnung T(t)
  - Dynamisch (alles zur Auswertungszeit)
  - Statisch (alles vorausberechnet)
  - Selektion zwischen Besuchsfolgen (teilweise statisch, teilweise dynamisch)
- *Auswertungsphase*: Attributauswertung gemäß T(t)
  - Bedarfsgetrieben (demand-driven, „top-down“)
    - Auswerter wird mit Menge von Attributexemplaren aufgerufen, deren Werte erwünscht sind (maximale Elemente bezüglich Auswertungsordnung)
    - Dieser ruft dann (rekursiv) die Auswertung der benötigten Argumente auf
  - Datengetrieben (data-driven, „bottom-up“)
    - Attributauswertung beginnt mit der Auswertung der Attributexemplare, die von keinen anderen abhängen (minimale Elemente bezüglich Auswertungsordnung)
    - Exemplar wird immer erst dann ausgewertet, wenn alle Argumente bereits berechnet sind
- Für beide Phasen: Varianten (die spezielle Klassen von Auswertern festlegen)

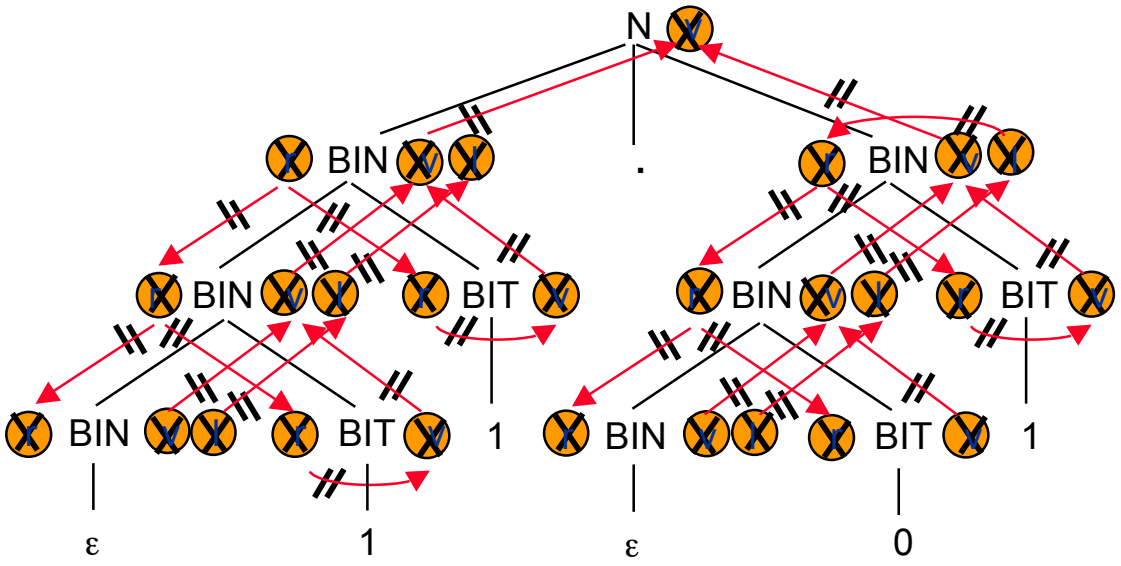
beginnt stets mit min./max. Elementen (= Quellen/Senken im Abhängigkeitsgraph)

Attributauswertung (3/4)

Dynamische Bestimmung der Auswertungsordnung

- **Topologisches Sortieren** des individuellen Abhängigkeitsgraphen Dt(t) ergibt totale Ordnung T(t) auf V(t)
- Problem: platz- und zeitaufwendig

Beispiel



r_1, v_{111}, l_{111}, v_{311}, l_{311}, v_{312} , r_{11}, r_{12}, l_{11}, v_{31}, l_{31} , r_{111}, r_{112}, l_1, v_{12}, l_3 , v_{111}, r_3 , v_{11}, r_{31}, r_{32} , v_1, r_{311}, r_{312}, v_{32} , v_3, v_0

Attributauswertung (4/4)

(statisch bestimmbare) **induzierte Auswertungsordnung**

- Bei spezieller Klasse von AGs (l-geordnete Attributgrammatiken) sind die individuellen Abhängigkeitsgraphen aller Bäume verträglich mit totalen Ordnungen auf den Attributmengen der Nichtterminale
- Ausgehend von diesen totalen Ordnungen auf den Attributmengen der Nichtterminale können „Besuchsfolgen“ (= totale Ordnungen auf den Mengen der Attributvorkommen für beliebige Produktionen) konstruiert werden
- Diese induzieren dann eindeutig eine Auswertungsordnung T(t) für jeden Baum t (d.h. zur Auswertungszeit: Strategiephase bereits erledigt)

Selektion zwischen Besuchsfolgen

- Problem
  - Produktionen, denen in verschiedenen Baumkontexten verschiedene Besuchsfolgen zugeordnet werden müssen
- Lösung
  - Statisch: Berechnung der Menge aller Besuchsfolgen für jede Produktion
  - Dynamisch: Auswahl der richtigen Besuchsfolge für jede Anwendung einer Produktion

L-attributierte Grammatiken (1/2)

Spezielle Attributgrammatikklassen

- Stark eingeschränkt
- Wegen Zeit- und Platzeffizienz trotzdem praktisch interessant

Charakteristika

- Attributauswertung parallel zur Syntaxanalyse (parsergesteuert)
- Kellerartige Verwaltung der Attributwerte
  (damit Aufbau des Syntaxbaums zur späteren Attributauswertung überflüssig)

L-attributierte Grammatiken

- Oberklasse aller Parser-gesteuert auswertbaren Attributgrammatiken
  (ererbte Attributvorkommen auf der rechten Seite einer Produktion hängen nur von Vorkommen ab, die „links von ihnen“ stehen)
- Enthält alle Grammatiken, in denen in jedem Syntaxbaum die Attribute in einem links-rechts-Tiefendurchlauf ausgewertet werden können
- Spezialfall: S-attributiert = hat nur abgeleitete Attribute

Beispiele

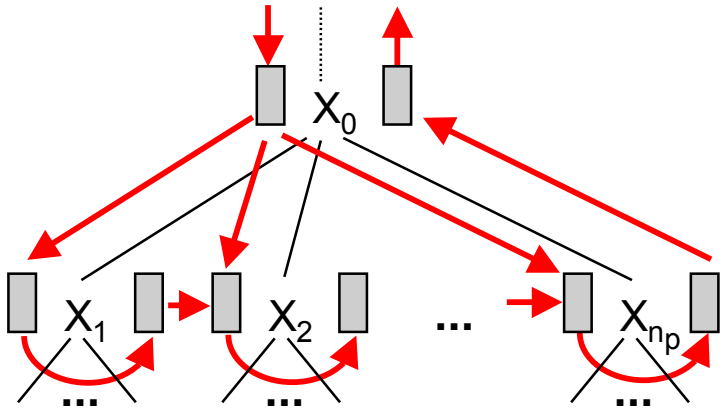
- AG_1 und AG_2 (AG_1 ist sogar S-attributiert)

Typberechnung für arithmetische Ausdrücke

L-attributierte Grammatiken (2/2)

Attributauswertung in einer L-attributierten Grammatik

```
program L-AA
proc visit (n: node)
begin
    case prod(n) of
        :
    p:  begin
            eval(Inh(X1)); visit(n1);
            eval(Inh(X2)); visit(n2);
                :
            eval(Inh(Xnp)); visit(nnp);
            eval(Syn(X0));
        end;
    endcase
end;
begin
    visit(ε)
end.
```



- Aktionen für Parser-gesteuerte Attributauswertung im LL-Parser
  - eval(Inh(X)) bei Beginn der Analyse eines Wortes für X (d.h. bei der Expansion)
  - eval(Syn(X)) bei Ende der Analyse für X (d.h. bei der Reduktion)
  - get(Syn(X)) beim Lesen eines Terminals X

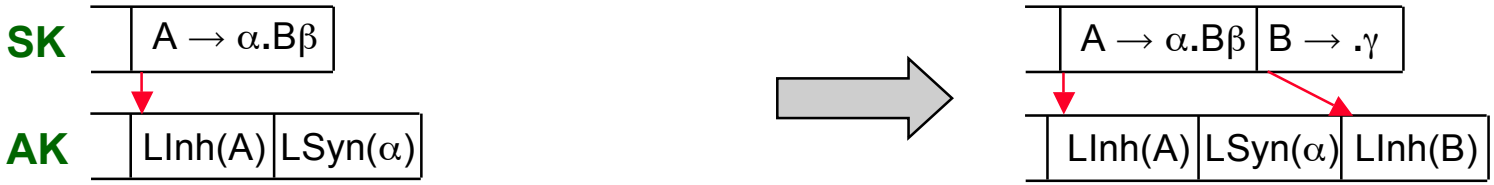
LL-attributierte Grammatiken

LL-attributierte Grammatik

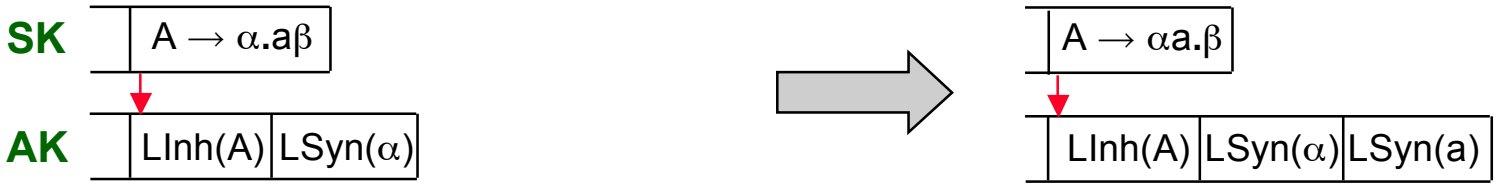
- AG ist **LL-attributiert** ⇔ AG ist L-attributiert + zugrunde liegende kfG ist LL-Grammatik

Implementierung von LL-attributierten Grammatiken

- Expansion eines Nichtterminals B

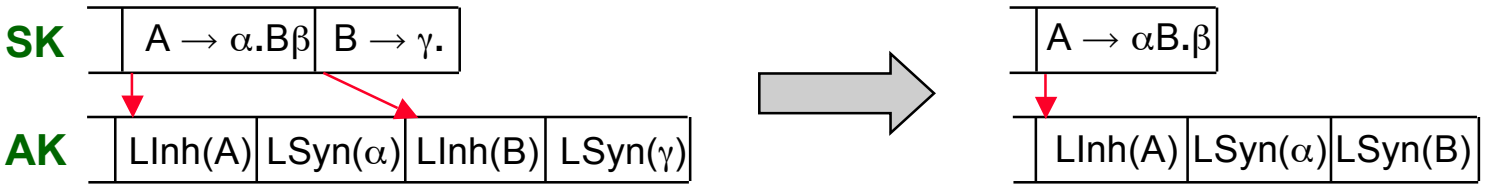


- Lesen eines Terminalsymbols a



Linh/Lsyn bezeichnen die (geordnete) Liste der Inh/Syn-Attribute (mit relativer Adressierung) der einzelnen Attribute

- Reduktion gemäß B → γ



LR-attributierte Grammatiken (1/2)

LR-attributierte Grammatiken

- Auswahl einer Produktion
  - LL: bei Expansion
  - LR: bei Reduktion
- Deshalb im LR-Parser
  - Attributauswertungsaktionen an Reduktionen + Leseaktionen des LR-Parsers koppeln
  - Damit verbunden: Kellern der Werte der abgeleiteten Attribute (im Attributkeller)
- AG ist **LR-attributiert** ⇔ AG ist L-attributiert + zugrunde liegende kfG ist LR-Grammatik

Allgemeine Probleme (mit ererbten Attributen)

- Oberes Baumfragment, in dem die Transportwege für ererbte Attributwerte liegen, ist noch nicht bekannt (für L-attributierte Grammatik allerdings bekannt: alle Attribute, von denen ein ererbtes Attribut eines Nichtterminals abhängt, sind bereits berechnet)
- Bei Beginn der Analyse eines Wortes für ein Nichtterminal müssen die Werte seiner ererbten Attribute verfügbar gemacht werden; dieser Zeitpunkt kann vom LR-Parser evtl. nicht festgestellt werden — *Mehrere Items im betreffenden Zustand*
- Adressierung der Attribute positiv relativ zu einem Basiszeiger aus dem Syntaxkeller funktioniert nicht (aus demselben Grund) — *Parser weiß nicht, wann Basiszeiger gesetzt werden muss*

LR-attributierte Grammatiken (2/2)

Lösung: Grammatiktransformation

- Ziel: Berechnung ererbter Attribute an Reduktion koppeln
- Vorgehensweise
  - Vor Nichtterminale X_i (in den rechten Produktionsseiten) neues Nichtterminal N mit
    - Produktion $N \rightarrow \varepsilon$
    - Attributregeln „LSyn(N) = Linh(X_i)“ einfügen
- Damit (bei Reduktion von ε zu N)
  - Berechnung der Attribute in Linh(X)
  - Ablage der Werte oben auf dem Attributkeller

Für jedes der jeweiligen Attribute

Verbleibende Probleme

- Zu viele neue Nichtterminale (Kellerplatzverbrauch zu hoch)
- Evtl. Verlust der LR-Eigenschaft

(verbesserte) Lösung

- Nur dort ε-Nichtterminal einfügen, wo die Werte der ererbten Attribute eines Nichtterminals nicht das obere Ende des Attributkellers bilden
- Ist bereits ein Präfix der Liste der Werte (der ererbten Attribute) oben auf dem Attributkeller, dann nur soviel berechnen, dass Präfix zur ganzen Liste ergänzt wird

Attributierung in YACC/Bison

In Bison

- Vordefinierte S-Attributierung (genau ein abgeleitetes Attribut zu jedem Grammatiksymbol)
- Grammatikregeln geben Produktionen und damit assoziierte semantische Aktionen an

Beispiel (Grammatik für variablenfreie, arithmetische Ausdrücke mit semantischen Regeln)

```
expr NUM                        {$$ = $1 }
     | exp '+' exp              {$$ = $1 + $3;   }
     | exp '-' exp              {$$ = $1 - $3;   }
     | exp '*' exp              {$$ = $1 * $3;   }
     | exp '/' exp              {$$ = $1 / $3;   }
     | '-' exp %prec NEG        {$$ = - $2;      }
     | exp '^' exp              {$$ = pow($1, $3);}
     | '('exp')'                {$$ = $2;  }
;
```

Dabei

- `$$`: Vorkommen des Attributs auf der linken Produktionsseite
- `$i`: Vorkommen des Attributs beim i-ten Symbol der rechten Seite