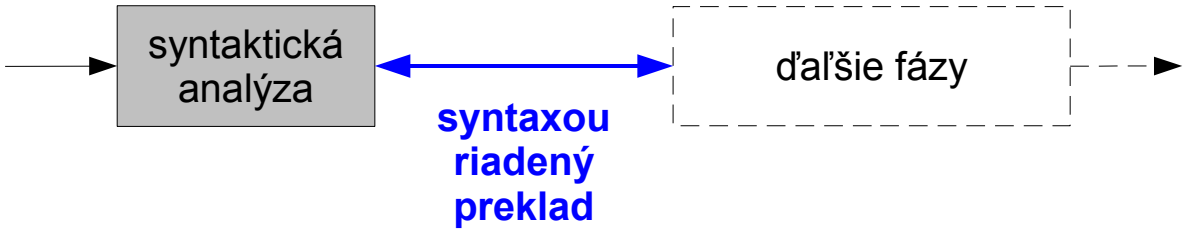
KOMPILÁTORY: Syntaxou riadený preklad

Jana Dvořáková
dvorakova@dcs.fmph.uniba.sk

Čo je syntaxou riadený preklad?

- Celý kompilátor je riadený procesom parsovania
- Syntaxou riadený preklad = prepojenie syntaktickej analýzy s nasledujúcimi fázami kompilácie



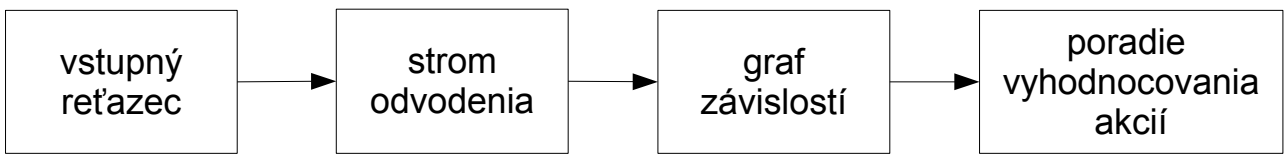
- K pravidlám gramatiky sú priradené akcie
  - Generovanie kódu
  - Ukladanie informácií do tabuľky symbolov
  - Generovanie chybových hlásení
  - Kontrola konzistencie typov
  - ..a ďalšie
- Notácia
  1. Syntaxou riadená definícia
     - Abstraktná notácia
  2. Prekladová schéma
     - Podobná syntaxou riadenej definícií, ale viac implementačných detailov (poradie vykonania akcií)

Realizácia akcií

1. Počas parsovania
   - Jeden prechod, používa sa v kompilátoroch
   - Bez explicitnej konštrukcie stromu odvodenia
   - Možné pre špeciálne typy syntaxou riadených definícií

     - Napr. L-atribútové definície zahŕňajú prakticky všetky syntaxou riadené definície, ktoré sa dajú realizovať bez explicitnej konštrukcie stromu odvodenia

2. Prechádzaním explicitne zostrojeného stromu odvodenia



Syntaxou riadená definícia (SRD)

- Abstraktná špecifikácia syntaxou riadeného prekladu
- Vznikne rozšírením CF gramatiky vstupného jazyka o
  1. *Atribúty*
     - Priradené k symbolom gramatiky

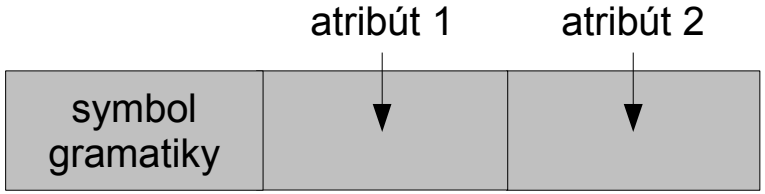
       $X.x$ = atribút x symbolu X
     - Nesú ďaľšiu informáciu o danom symbole
       - Hodnota atribútu: reťazec, číslo, typ, miesto v pamäti,..
  2. *Sémantické pravidlá*
     - Priradené k pravidlám gramatiky

       $B.b := f(C_1.c_1, \ldots, C_k.c_k)$
     - Slúžia pre výpočet atribútov symbolov v rámci jedného pravidla gramatiky
     - Môžu obsahovať funkcie s bočným efektom (napr. vypísanie hodnoty na výstup, zmena hodnoty globálnej premennej)

       SRD bez funkcií s bočným efektom = atribútová gramatika

Atribúty

- Vrchol v strome odvodenia = záznam s poľami



  - Symbol gramatiky je v prvom poli záznamu a je návestím vcholu
  - Atribúty sú uložené v ďalších poliach

- Typy atribútov
  1. Syntetizované atribúty
     - Hodnoty sa vypočítajú podľa hodnôt atribútov detí
     - V praxi často používané
  2. Dedičné atribúty
     - Hodnoty sa vypočítajú podľa hodnôt atribútov rodičov a súrodencov
     - Vhodné na vyjadrenie závislosti konštrukcií na kontexte, v ktorom sa nachádzajú

- Ak $B.b := f(C_1.c_1, \ldots, C_k.c_k)$ hovoríme, že $B.b$ je *závislý* na $C_1.c_1, \ldots, C_k.c_k$
- Strom odvodenia s vypočítanými hodnotami atribútov sa nazýva *anotovaný strom odvodenia*
- Terminály majú iba syntetizované atribúty a poč. symbol gramatiky má iba dedičné atribúty

Syntaxou riadená definícia

Príklad 1 - kalkulačka

Pravidlo gramatiky	Sémantické pravidlo
$L \to E\,\mathbf{n}$	$print(E.val)$ ← funkcia s bočným efektom
$E \to E_1 + T$	$E.val := E_1.val + T.val$
$E \to T$	$E.val := T.val$
$T \to T_1 * F$	$T.val := T_1.val \times F.val$
$T \to F$	$T.val := F.val$
$F \to (\,E\,)$	$F.val := E.val$
$F \to \mathbf{digit}$	$F.val := \mathbf{digit}.lexval$ ← hodnota atribútu je poslaná lexikálnym analyzátorom

- $X.val$ je syntetizovaný atribút pre $X \in \{E, T, F\}$ s celočíselnou hodnotou

SRD iba so syntetizovanými atribútmi sa nazýva S-atribútová definícia a vždy sa dá vyhodnotiť na strome odvodenia prechodom zdola-nahor

Syntaxou riadená definícia

Príklad 2 - deklarácie

Pravidlo gramatiky	Sémantické pravidlo
$D \rightarrow T\ L\ ;$	$L.in := T.type$
$T \rightarrow$ **int**	$T.type := integer$
$T \rightarrow$ **real**	$T.type := real$
$L \rightarrow L_1\ ,$ **id**	$L_1.in := L.in$
	$addtype($**id**$.entry, L.in)$
$L \rightarrow$ **id**	$addtype($**id**$.entry, L.in)$

- $L.in$ je dedičný atribút
- $T.type$ je syntetizovaný atribút

Nedá sa vyhodnotiť jednoducho ako S-atribútové definície

Graf závislostí

- Znázorňuje závislosti medzi atribútami
- Orientovaný graf $D = (V, E)$
  - V - množina vrcholov, atribúty symbolov gramatiky
  - E - množina hrán, $X.x \rightarrow Y.y \in E$ ak $Y.y$ je závislý na $X.x$

```
for každý vrchol n v strome odvodenia do
    for každý atribút a symbolu gramatiky v n do
        vytvor vrchol pre a;
for každý vrchol n v strome odvodenia do
    for každé sémantické pravidlo b := f(c_1, ..., c_k) asociované s pravidlom v n do
        for i := 1 to k do
            pridaj hranu z c_i do b
```

- Topologické triedenie D
  - Usporiadanie vrcholov $m_1, \ldots, m_k$, také, že platí:

    Ak existuje hrana $m_i \rightarrow m_j$ potom m_i je pred m_j.

  - Získame platné poradie vyhodnocovania atribútov (ak D nemá cyklus)

Metódy vyhodnocovania atribútov

1. Metódy stromu odvodenia
   - Strom odvodenia sa zostrojí explicitne, nájde sa graf závislostí a topologickým triedením sa získa poradie vyhodnocovania atribútov
   - V čase kompilácie
2. Metódy analýzy pravidiel
   - Poradie vyhodnocovania sa určí statickou analýzou sémantických pravidiel
   - V čase konštrukcie kompilátora
3. "Nedbanlivé"metódy *
   - Poradie vyhodnocovania sa určí bez ohľadu na sémantické pravidlá
   - Ak sa preklad vykonáva počas parsovania, poradie je určené parsovacou metódou
   - V čase konštrukcie kompilátora

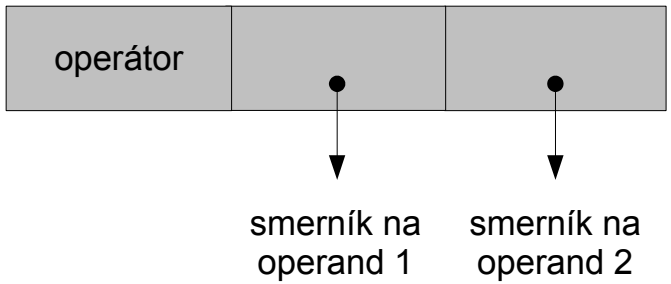
Syntaktický strom

- Zjednodušený strom odvodenia
  - Operátory a kľúčové slová sú vnútornými vrcholmi
  - Vetvy odvodené reťazcovými pravidlami (chain rules) sú zredukované
- Využíva sa ako prechodná reprezentácia kompilovaného programu alebo jeho častí
  - Syntaxou riadený preklad môže byť založený na strome odvodenia alebo syntaktickom strome
  - Rovnaký prístup, atribúty sú priradené k uzlom stromu a ten sa anotuje
  - Výhody syntaktického stromu
    - Gramatika vhodná na parsovanie nie vždy reprezentuje prirodzenú hierarchickú štruktúru vstupného programovacieho jazyka
    - Parsovacia metóda obmedzuje poradie sprístupnenia uzlov stromu odvodenia a toto poradie nie je vždy vhodné pre preklad

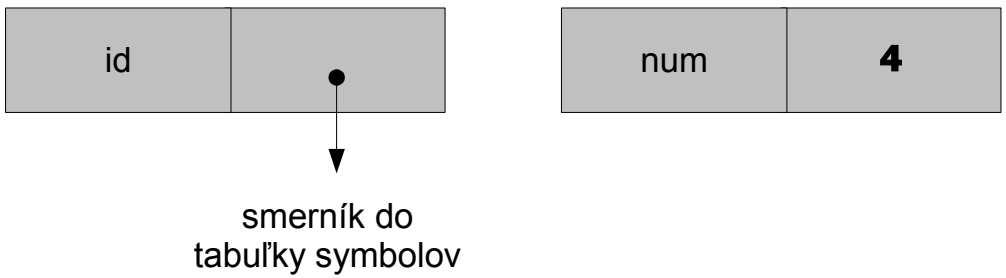
Syntaktický strom pre výrazy

- Funkcie na konštrukciu vrcholov:
  1. *mknode*(*op*, *left*, *right*)
  2. *mkleaf*(**id**, *entry*)
  3. *mkleaf*(**num**, *val*)

  $\Rightarrow$ Vrátia smerník na skonštruovaný vrchol
- Vrcholy pre operátory sú implementované ako záznamy s niekoľkými poliami



  - Operátor je návestím vrcholu
  - Pri preklade sú v zázname ešte ďalšie polia pre atribúty
- Vrcholy pre identifikátory a číselné konštanty



Syntaktický strom pre výrazy

Syntaxou riadená definícia

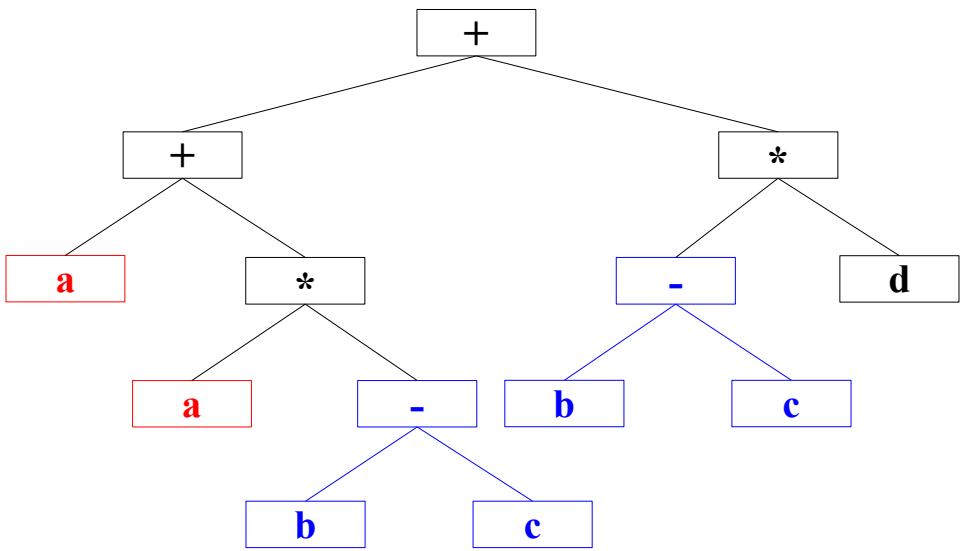
Pravidlo gramatiky	Sémantické pravidlo
$E \rightarrow E_1 + T$	$E.nptr := mknode('+', E_1.nptr, T.nptr)$
$E \rightarrow E_1 - T$	$E.nptr := mknode('-', E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr := T.nptr$
$T \rightarrow T_1 * F$	$T.nptr := mknode('\times', T_1.nptr, F.nptr)$
$T \rightarrow (E)$	$T.nptr := E.nptr$
$T \rightarrow \mathbf{id}$	$T.nptr := mkleaf(\mathbf{id}, \mathbf{id}.entry)$
$T \rightarrow \mathbf{num}$	$T.nptr := mkleaf(\mathbf{num}, \mathbf{num}.val)$

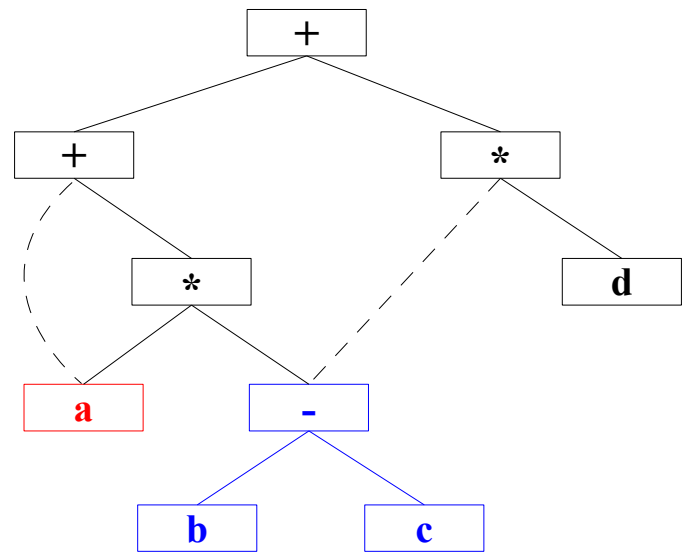
- $X.nptr$ je syntetizovaný atribút pre $X \in \{E, T, F\}$ a obsahuje smerník na príslušný vrchol

Syntaktický strom vs. DAG

- Spoločné podvýrazy sú reprezentované iba jedným podstromom
- Pri pridávaní nového vrchola sa kontroluje, či už identický vrchol neexistuje

Výraz: $a + a * (b - c) + (b - c) * d$





Implementácia SRD pri parsovaní

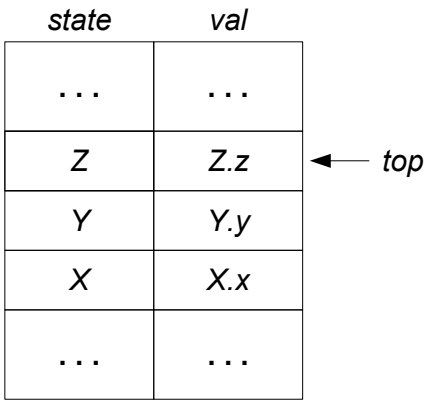
- Zostrojiť automatický prekladač pre ľubovoľnú SRD môže byť ťažké
- Máme veľké triedy, pre ktoré to je jednoduché
  1. S-atribútové definície
     - Obsahujú iba syntetizované atribúty
  2. L-atribútové definície
     - Obsahujú syntetizované aj dedičné atribúty, ale sú tu urcčité obmedzenia pre výpočet dedičných atribútov
     - Zahŕňajú všetky SRD, ktoré sa implementujú bez explicitnej konštrukcie stromu odvodenia
- S-atribútové definície $\subsetneq$ L-atribútove definície
- Obe triedy sa dajú implementovať pri parsovaní zhora-nadol aj zdola-nahor
  - Obmedzenia na použitú gramatiku

S-atribútová definícia

Implementácia pri parsovaní zdola-nahor (1)

- Hodnoty atribútov sú uložené v špeciálnych poliach v zásobníku parsera a sú asociované so symbolmi gramatiky (resp. stavmi)
  - Ak symbol nemá atribút, hodnota je nedefinovaná
- Zásobník s miestom pre jeden atribút:

Implementácia - dvojica polí



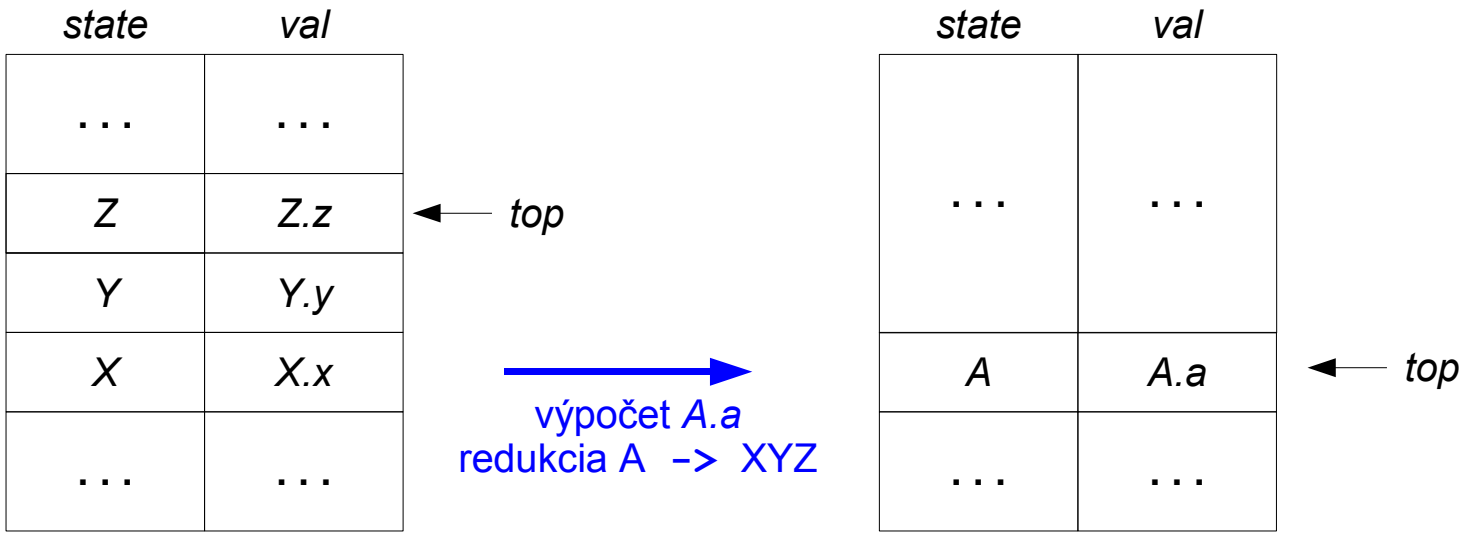
1. *state* obsahuje smerníky (indexy) do LR(1) parsovacej tabuľky
   - Symbol gramatiky je implicitne v stave, netreba si ho pamätať
   - Pre jednoduchosť uvádzame symbol namiesto príslušného stavu
2. *val* obsahuje hodnoty atribútu
   - *val*[*i*] = hodnota atribútu pre symbol v *state*[*i*]

- Hodnoty syntetizovaných atribútov sú vypočítané vždy pred redukciou z atríbútov uložených v zásobníku (= atribútov symbolov z pravej strany pravidla)

S-atribútová definícia

Implementácia pri parsovaní zdola nahor (2)

Príklad: $A \rightarrow XYZ$, $A.a = f(X.x, Y.y, Z.z)$



- $Z.z := val[top]$
- $Y.y := val[top - 1]$
- $X.x := val[top - 2]$

- $top := top - 2$
- $val[top] := A.a$

S-atribútová definícia

Implementácia pri parsovaní zdola nahor (3)

Pravidlo gramatiky	Fragment kódu
$L \rightarrow E\ \mathbf{n}$	$print(val[top])$
$E \rightarrow E_1 + T$	$val[ntop] := val[top-2] + val[top]$
$E \rightarrow T$	
$T \rightarrow T_1 * F$	$val[ntop] := val[top-2] \times val[top]$
$T \rightarrow F$	
$F \rightarrow (\ E\ )$	$val[ntop] := val[top-1]$
$F \rightarrow \mathbf{digit}$	

- S redukciami v LR parseri sú asociované fragmenty kódu
  - Získame ich zo sémantických pravidiel nahradením atribútov pozíciami v poli *val*
  - Vykonajú sa tesne pred redukciou
- Premenné *top* a *ntop*
  - *top* je aktuálny vrch zásobníka, *ntop* je nový vrch zásobníka
  - Postup pri redukcii podľa pravidla $A \rightarrow \beta$, kde $|\beta| = r$
    1. *ntop* sa nastaví na $top - r + 1$
    2. Vykoná sa fragment kódu a redukcia
    3. *top* sa nastaví na *ntop*

L-atribútová definícia

- Syntaxou riadená definícia je L-atribútová, ak pre ľubovoľné pravidlo $A \to X_1, X_2 \ldots X_n$ platí, že každý dedičný atribút symbolu z pravej strany X_j je závislý iba na
  1. atribútoch symbolov $X_1, X_2, \ldots X_{j-1}$, ktoré sú v pravidle naľavo od X_j a
  2. dedičných atribútoch symbolu A.

- SRD, ktorá nie je L-atribútová

Pravidlo gramatiky:	Sémantické pravidlo:	
$A \to LM$	$L.i := l(A.i)$	
	$M.i := m(L.s)$	
	$A.s := f(M.s)$	
$A \to QR$	$R.i := r(A.i)$	
	$Q.i := q(R.s)$	← Nezodpovedá definícii
	$A.s := f(Q.s)$	

- Pozn.: Každá S-atribútová definícia je aj L-atribútová

L-atribútová definícia

Vyhodnotenie atribútov

- Atribúty môžu byť vyhodnotené prechádzaním stromu do hĺbky (depth-first order)

```
procedure dfvisit(n:node);
begin
    for každé dieťa m vrcholu n zľava doprava do begin
        vyhodnoť dedičné atribúty vrcholu m;
        dfvisit(m)
    end;
    vyhodnoť syntetizované atribúty vrcholu n
end
```

- Prirodzený spôsob prechádzania stromu odvodenia
- Implementácia L-atribútovej definície pri parsovaní
  1. Parsovanie zhora-nadol
     - Všetky SRD založené na LL(1) gramatikách
  2. Parsovanie zdola-nahor
     - Všetky SRD založené na LL(1) gramatikách, veľká časť SRD založených na LR(1) gramatikách

Prekladové schémy

- Ďalší spôsob špecifikácie syntaxou riadeného prekladu
- Konkrétnejšia ako SRD
  - Špecifikuje aj poradie vyhodnocovania atribútov
  - Sémantické akcie sú uzátvorkované pomocou {} a vložené medzi symboly pravých strán pravidiel
- Príklad
  - Prekladová schéma, ktorá prekladá infixové výrazy so sčítaním a odčítaním na ekvivalentné postfixové výrazy

$$
\begin{aligned}
E &\rightarrow T\ R \\
R &\rightarrow \textbf{addop}\ T\ \{\mathit{print}(\textbf{addop}.\mathit{lexeme})\}\ R_1 \mid \varepsilon \\
T &\rightarrow \textbf{num}\ \{\mathit{print}(\textbf{num}.\mathit{val})\}
\end{aligned}
$$

  - **addop**.*lexeme* môže byť '+' alebo '-'
  - Gramatika vznikla odstránením ľavej rekurzie z povestnej gramatiky pre výrazy

Obmedzenia pre prekladové schémy

- Akcia nesmie odkazovať na nevypočítaný atribút

- Prekladové schémy pre S-atribútové definície
  - Jednoduché riešenie - akcie sú umiestnené na konci pravidiel
- Prekladové schémy pre L-atribútové definície
  1. Dedičný atribút symbolu na pravej strane pravidla musí byť vypočítaný v akcii naľavo od neho.
  2. Akcia nesmie odkazovať na syntetizovaný atribút symbolu napravo.
  3. Syntetizovaný atribút neterminálu na ľavej strane pravidla môže byť vypočítaný až potom ako boli vypočítané všetky atribúty na ktorých je závislý.
     - Príslušná akcia pre jeho vypočítanie je zvyčajne lokalizovaná na konci pravej strany pravidla

  - Príklad "zlej"prekladovej schémy

$$S \rightarrow A_1\ A_2 \quad \{A_1.in := 1; A_2.in := 2;\}$$
$$A \rightarrow a \quad \{print(A.in)\}$$

L-atribútová definícia

Implementácia pri prediktívnom parsovaní

- Implementácia je možná iba pre L-atribútové definície (prekladové schémy *) založené na LL(1) gramatikách, ktoré okrem iného, nemôžu obsahovať ľavú rekurziu
- Existuje metóda na odstránenie ľavej rekurzie priamo z prekladovej schémy
  - Rozšírenie algoritmu odstránenia ľavej rekurzie z CF gramatiky
  - Dá sa aplikovať na prekladové schémy so syntetizovanými atribútmi
- Ak už máme prekladovú schému založenú na LL(1) gramatike, existuje algoritmus pre konštrukciu prediktívneho syntaxou riadeného prekladača

Odstránenie ľavej rekurzie z prekl. schémy

- Vstupná prekladová schéma:

$$\begin{array}{lcll} A & \rightarrow & A_1\ Y & \{A.a := g(A_1.a, Y.y)\} \\ A & \rightarrow & X & \{A.a := f(X.x)\} \end{array}$$

- Po odstránení ľavej rekurzie z CF gramatiky získame novú gramatiku:

$$\begin{array}{lcl} A & \rightarrow & X\ R \\ R & \rightarrow & Y\ R \mid \varepsilon \end{array}$$

- Po prepísaní sémantických akcií získame výslednú prekl. schému:

$$\begin{array}{lcll} A & \rightarrow & X & \{R.i := f(X.x)\} \\ & & R & \{A.a := R.s\} \\ R & \rightarrow & Y & \{R_1.i := g(R.i, Y.y)\} \\ & & R_1 & \{R.s := R_1.s\} \\ R & \rightarrow & \varepsilon & \{R.s := R.i\} \end{array}$$

Odstránenie ľavej rekurzie z prekl. schémy

Príklad

$$
\begin{array}{lcll}
E & \to & E_1 + T & \{E.val := E_1.val + T.val\} \\
E & \to & E_1 - T & \{E.val := E_1.val - T.val\} \\
E & \to & T & \{E.val := T.val\} \\
T & \to & (E) & \{T.val := E.val\} \\
T & \to & \textbf{num} & \{T.val := \textbf{num}.val\}
\end{array}
$$

$$
\begin{array}{lcll}
E & \to & T & \{R.i := T.val\} \\
 & & R & \{E.val := R.s\} \\
R & \to & + & \\
 & & T & \{R_1.i := R.i + T.val\} \\
 & & R_1 & \{R.s := R_1.s\} \\
R & \to & - & \\
 & & T & \{R_1.i := R.i - T.val\} \\
 & & R_1 & \{R.s := R_1.s\} \\
R & \to & \varepsilon & \{R.s := R.i\} \\
T & \to & (& \\
 & & E & \\
 & &) & \{T.val := E.val\} \\
T & \to & \textbf{num} & \{T.val := \textbf{num}.val\}
\end{array}
$$

Konštrukcia prediktívneho prekladača

Vstup. Syntaxou riadená prekladová schéma založená na gramatike vhodnej pre prediktívne parsovanie.

Výstup. Kód prediktívneho syntaxou riadeného prekladača.

1. Pre každý neterminál A vytvor funkciu, ktorá má formálny parameter pre každý dedičný atribút A a vracia hodnoty syntetizovaných neterminálov A. Funkcia pre A má lokálnu premennú pre každý atribút každého symbolu, ktorý sa vyskytuje v pravidle pre A.
2. Kód pre neterminál A sa rozhoduje, ktoré pravidlo aplikovať podľa aktuálneho vstupného symbolu.
3. Kód asociovaný s každým pravidlom vykonáva nasledovné akcie.
   - (i) Pre token X so syntetizovaným atribútom x, ulož hodnotu x do premennej deklarovanej pre $X.x$. Potom zavolaj procedúru $match(X)$ a posuň sa na vstupe.
   - (ii) Pre neteminál B, vygeneruj priradenie tvaru $c := B(b_1, b_2, \ldots, b_k)$, kde $b_1, b_2, \ldots, b_k$ sú premenné pre dedičné atribúty neterminálu B, c je premenná pre syntetizovaný atribút net. B a B je volanie funkcie pre net. B.
   - (iii) Pre akciu, kopíruj kód priamo do parsera, pričom každú referenciu na atribút nahraď asociovanou premennou.

Syntaxou riadený preklad pri parsovaní

Zhrnutie

1. Implementácia S-atribútových definícii
   - Mali sme algoritmus pri parsovaní zdola-nahor
   - Dajú sa implementovať aj pri parsovaní zhora-nadol

     - Odstránenie ľavej rekurzie z prekladovej schémy a zostrojenie prediktívneho prekladača
2. Implementácia L-atribútových definícií
   - Mali sme algoritmus pri parsovaní zhora-nadol
   - Dajú sa implementovať aj pri parsovaní zdola-nahor

     - Rozšírenie algoritmu pre spracovanie S-atribútových definícií
   - V oboch prípadoch je obmedzená množina CF gramatík, na ktorej je SRP založený

     - Prekladač zdola-nahor je silnejší