Strong LL(k) parsing

Computation of $First_k$.

Let $k \geq 0$, A be a set and $\alpha, \beta \in A^*$. Then we define

$$\alpha \oplus_k \beta = k{:}\alpha\beta.$$

$\therefore\ \oplus_k{:}A^* \times A^* \rightarrow A^{\leq k}$ binary operation on A.

where $A^{\leq k} = A^0 \cup A^1 \cup A^2 \ldots \cup A^k = \{\varepsilon\} \cup A \cup A^2 \ldots \cup A^k$

$\leftrightarrow \$^k \cup A\$^{k-1} \cup A^2\$^{k-2} \ldots \cup A^k.$

We define languages and $First_k$ of $\alpha \in (N \cup \Sigma)^*$ and $K \subseteq (N \cup \Sigma)^*$.

$$L(\alpha) = \{w \in \Sigma^* \mid \alpha \Rightarrow^* w\}$$

$$First_k(\alpha) = k{:}L(\alpha)$$

$$L(K) = \{w \in \Sigma^* \mid \alpha \in K, \alpha \Rightarrow^* w\}$$

$$First_k(K) = k{:}L(K)$$

Divide and conquer on the computation of $First_k$.

1. $First_k: 2^{\Sigma^*} \to 2^{\Sigma^{\leq k}}$. Let $L_1, L_2 \subseteq \Sigma^*$. Then

$$First_k(L_1L_2) = First_k(L_1) \oplus_k First_k(L_2) = k{:}L_1L_2.$$

2. $First_k: (N \cup \Sigma)^* \to 2^{\Sigma^{\leq k}}$. Let $\alpha = X_1 \ldots X_n, 1 \leq {}^{\forall}i \leq n, X_i \in N \cup \Sigma$.

$$First_k(\alpha) = First_k(X_1 \ldots X_n)$$
$$= First_k(X_1) \oplus_k First_k(X_2) \oplus_k \ldots \oplus_k First_k(X_n).$$

3. $First_k: N \cup \Sigma \to 2^{\Sigma^{\leq k}}$.

$First_k(a) = \{a\}$, if $a \in \Sigma$. **basis**

$First_k(A) = \{x \in First_k(\alpha) \mid A \to \alpha \in P\}$ **recursion**

We also define $Follow_k: N \to 2^{\Sigma^{\leq k}}$. Let $A \in N$. Then

$$Follow_k(A) = \{k{:}z \in \Sigma^* \mid S \Rightarrow^* xAz\}$$

Guess-verify parser $M = (N \cup \Sigma, \Sigma, \Gamma, S, \{\varepsilon\}, \$, |)$

for $G = (N, \Sigma, P, S)$

$\forall A \to \omega \in P$, $\quad A \,|\, \to \omega^R \,| \in \Gamma$, $\quad$ ***guess*** A as α.

$\forall a \in \Sigma$, $\quad a \,|\, a \to \;| \in \Gamma$, $\quad$ ***verify*** $a \in \Sigma$.

$$.S \Rightarrow_{lm}^{*} x.A\gamma \Rightarrow_{lm} x.\beta\gamma \Rightarrow_{lm}^{*} xy.\gamma \Rightarrow_{lm}^{*} xyz$$

$$\$S \,|\, xyz\$ \Rightarrow^{*} \$\gamma^R A \,|\, yz\$ \Rightarrow \$\gamma^R\beta^R \,|\, yz\$ \Rightarrow^{*} \$\gamma^R \,|\, z\$ \Rightarrow^{*} \$ \,|\, \$.$$

Nondeterminism

If $A \to \alpha \mid \beta \in P, \alpha \neq \beta$.

guess A as α or β $\quad$ ***nondeterministic***

Adding k-lookahead symbols y where $|y| \leq k$.

$\forall A \to \alpha \in P$, $\quad A \,|\, y \to \alpha^R \,|\, y \in \Gamma$, *where y* $\in First_k(\alpha) \oplus_k Follow_k(A)$

Guess *A as* α **on lookahead** *y only.*

We define $LA_k(A \to \alpha) = First_k(\alpha) \oplus_k Follow_k(A)$.

Guess and verify parser with adding k-lookahed symbols, $LA_k(A \to \alpha)$*,*
is called **strong LL(k)** *parser.*

If $LA_k(A \to \alpha) \cap LA_k(A \to \beta) = \emptyset$.
deterministic guess *A as* α *or* β *on* $LA_k(A \to \alpha)$ *or* $LA_k(A \to \beta)$.

G is **strong LL(k) grammar***, if*
$^{\forall}A \to \alpha \mid \beta \in P, \alpha \neq \beta, LA_k(A \to \alpha) \cap LA_k(A \to \beta) = \emptyset$.
Strong LL(k) parser is **deterministic***, if and only if G is strong LL(k).*

Theorem *G is not LL(k), if G is left recursive.*

$A \rightarrow A\alpha \mid \beta$

A is left recursive.

Proof $First_k(A\alpha) \supseteq First_k(A)$

$\supseteq First_k(\beta)$

$\therefore LA_k(A \rightarrow A\alpha) \cap LA_k(A \rightarrow \beta) \supseteq First_k(\beta) \neq \emptyset$

Removal of left recursion

$A \rightarrow A\alpha \mid \beta$

$A \Rightarrow^* \beta\alpha^*$

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \varepsilon$

Left factrong

$A \rightarrow \alpha\beta \mid \alpha\gamma$

$A \rightarrow \alpha A'$

$A' \rightarrow \beta \mid \gamma$

LL(1) parsing

Let $T_1, T_2 \subseteq \Sigma^*$. Consider

$$T_1 \oplus_1 T_2 = First_1(T_1T_2)$$
$$= First_1(T_1), \text{ if } \varepsilon \notin T_1.$$
$$= First_1(T_1) \cup First_1(T_2), \text{ if } \varepsilon \in T_1, \varepsilon \in T_2,$$
$$= First_1(T_1) \cup First_1(T_2) - \{\varepsilon\}, \text{ if } \varepsilon \in T_1, \varepsilon \notin T_2 (otherwise).$$

$First_1: N \cup \Sigma \to 2^{\Sigma^{\leq 1}} = 2^{\{\varepsilon\} \cup \Sigma}$. $\qquad First_1 \leftrightarrow nullable \cup First.$

$nullable: N \cup \Sigma \to \{true, false\}$

$nullable(A) = true, \text{ iff } A \Rightarrow^* \varepsilon.$

$First: N \cup \Sigma \to 2^{\Sigma}.$

$First(a) = \{a\}, \text{ if } a \in \Sigma.$

$First(A) = \{a \in \Sigma | A \Rightarrow^* ax, x \in \Sigma^*\}$

Recursive formula for First(A), $A \in N$.

$First(A) \supseteq \{a \in \Sigma \mid A \to \alpha a \beta \in P, \alpha \Rightarrow^* \varepsilon\}$ **basis**

$\underline{First}(A) \supseteq \{a \in \underline{First}(B) \mid A \to \alpha B \beta \in P, \alpha \Rightarrow^* \varepsilon\}$ **recursion**

$Follow_1: N \to 2^{\Sigma^{\leq 1}} = 2^{\{\varepsilon\} \cup \Sigma}$.

$Follow_1: N \to 2^{\{\varepsilon\} \cup \Sigma}$.

$Follow_1(A) = \{1{:}y \in \{\varepsilon\} \cup \Sigma \mid S \Rightarrow^* xAy, x, y \in \Sigma^*\}$

$\varepsilon \in Follow_1(S)$.

*(We may add new **end marker** \$ in Σ and ε ↔ \$.)*

Recursive formula for Follow(A)

$Follow(S) \supseteq \{\varepsilon\}$ **basis**

$Follow(A) \supseteq \{a \in First(\beta) \mid B \to \alpha A \beta \in P\}$ **basis**

$\underline{Follow}(A) \supseteq \{a \in \underline{Follow}(B) \mid B \to \alpha A \beta \in P, \beta \Rightarrow^* \varepsilon\}$ **recursion**

We define $l \subseteq N \times N$ *and* $r \subseteq N \times N$.

$A\ l\ B$, *if* $A \rightarrow \alpha B\beta \in P$, $\alpha \Rightarrow^* \varepsilon$, *and (left dependency relation)*

$A\ r\ B$, *if* $B \rightarrow \alpha A\beta \in P$, $\beta \Rightarrow^* \varepsilon$. *(right dependency relation)*

Formula for First(A) and Follow(A)

$First(A) \supseteq \{a \in \Sigma |\ A \rightarrow \alpha a\beta \in P, \alpha \Rightarrow^* \varepsilon\}$ **basis**

$= \{a \in \Sigma |\ A\ l\ a\}$

$\underline{First}(A) \supseteq \{a \in \underline{First}(B) |\ A\ l\ B\}$ **recursion**

$Follow(S) \supseteq \{\varepsilon\}$ **basis**

$Follow(A) \supseteq \{a \in First(\beta) |\ B \rightarrow \alpha A\beta \in P\}$ **basis**

$\underline{Follow}(A) \supseteq \{a \in \underline{Follow}(B) |\ A\ r\ B\}$ **recursion**

Let A and B be two sets and $x, y \in A$, $R \subseteq A \times A$, *and* $f, g: A \to 2^B$.
Compute f for given R and g,

$f(x) \supseteq g(x)$ **basis**

$f(x) \supseteq f(y)$ *where* $x\ R\ y$ **recursion**

$\therefore\ f(x) \supseteq g(x) \cup \cup_{xRy} f(y)$

Nothing else is in f(x) **fixed point**

$f(x) = g(x) \cup \cup_{xRy} f(y)$

Then $f(x) =_S \{b \in g(y) \mid x\ R^*\ y\}$ **iteration**

Reachable vertices in the graph R.

depth-first search

topological order

Algorithm *Compute f(x) with g(x) and R.*

input $g: A \rightarrow 2^B$; $R \subseteq A \times A$. ***output***: $f: A \rightarrow 2^B$.

```
var S: stack of A; N: A → Depth.
function Trav(x: A, d: Depth);
    push x onto S; N(x) = d;                    f(x) := g(x);                |A|
    for y ∈ A where x R y do
        if (N(y) = 0) then Trav(y, d+1) fi;
        N(x) = min(N(x), N(y));                 f(x) := f(x) ∪ f(y) od       |R|
    if (N(x) = d) then repeat
        y := pop of S; N(y) :=Infinity;         f(y) := f(x) until (y = x) fi
    end function Trav

for x ∈ A do N(x) := 0;                         f(x) := ∅ od;
for x ∈ A where (N(x) = 0) do Trav(x, 1) od
```

LL(1) analysis.

*1. Remove **useless** symbols and productions, if any.*

*2. **Remove left recursion**, and **do left factoring**, if any.*

3. $\forall A \in N$*, compute **nullale**(A).*

4. $\forall A \in N$*, compute l-relation and **initial** First(A) in **basis** using nullable.*

5. $\forall A \in N$*, compute **First**(A) using l-relation and initial First(A) by traversing l-graph.*

4. $\forall A \in N$*, compute r-relation and **initial** Follow(A) in **basis** using nullable and First(*β*).*

5. $\forall A \in N$*, compute **Follow**(A) using r-relation and **initial** Follow(A) in **basis** by traversing r-graph.*

6. $\forall A \to \alpha \in P$*, compute* $\mathbf{LA_1}(A \to \alpha)$ *using First and Follow.*

7. $\forall A \to \alpha \mid \beta \in P, \alpha \neq \beta$*, if* $LA_1(A \to \alpha) \cap LA_1(A \to \beta) = \emptyset$*, LL(1); otherwise not LL(1).*

Example *Expression grammar*

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow a \mid (E)$

Removal of left recursion

	nullable	init.	First	init.	Follow
$E \rightarrow T E'$	no		a, (	<u>ε</u>,)	ε,)
$E' \rightarrow + T E' \mid \varepsilon$	yes	+	+		ε,)
$T \rightarrow F T'$	no		a, (	+	ε,), +
$T' \rightarrow * F T' \mid \varepsilon$	yes	*	*		ε,), +
$F \rightarrow a \mid (E)$	no	a, (	a, (	*	ε,), +, *

$LA_1(E' \rightarrow + T E') = \{+\}$ $LA_1(E' \rightarrow \varepsilon) = \{\varepsilon,)\}$

$LA_1(T' \rightarrow * F T') = \{*\}$ $LA_1(T' \rightarrow \varepsilon) = \{\varepsilon,), +\}$

$LA_1(F \rightarrow a) = \{a\}$ $LA_1(F \rightarrow (E)) = \{(\}$

$\therefore$ LL(1).

$LL_1PT: N \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^P$.

$LL_1PT(A, a) = \{A \rightarrow \alpha \mid a \in LA_1(A \rightarrow \alpha)\}$

Fact G is $LL(1)$, if $\forall A \in N, \forall a \in \Sigma, |LL_1PT(A, a)| \leq 1$.

LL_1PT	a	$+$	$*$	$($	$)$	ε
E	$E\rightarrow TE'$			$E\rightarrow TE'$		
E'		$E'\rightarrow +TE'$			$E'\rightarrow\varepsilon$	$E'\rightarrow\varepsilon$
T	$T\rightarrow FT'$			$T\rightarrow FT'$		
T'		$T'\rightarrow\varepsilon$	$T'\rightarrow *FT'$		$T'\rightarrow\varepsilon$	$T'\rightarrow\varepsilon$
F	$F\rightarrow a$			$F\rightarrow (E)$		

Algorithm *LL(1) parsing*

input $LL_1PT: N \times (\Sigma \cup \{\varepsilon\}) \rightarrow P$; *input string* $x \in \Sigma^*$.

var *Stack: stack of* $N \cup \Sigma$; $a \in \Sigma$; $y \in \Sigma^*$.

push *S* ***onto*** *Stack;* $a := 1{:}x$; $y := x{:}|x|\text{-}1$

repeat

  $X :=$ ***pop of*** *Stack;*

  if $(X \in N) \Rightarrow$

    if $(LL_1PT(X, a) = X \rightarrow \alpha) \Rightarrow$ ***push*** α *onto Stack;*

      Make subtree with root X and α*'s as children.*

    $|\ LL_1PT(X, a) = \emptyset \Rightarrow$ *syntax error* ***fi***

  $|\ (X \in \Sigma) \Rightarrow$ ***if*** $(X{=}a) \Rightarrow a := 1{:}y;\ y := y{:}|y|\text{-}1$

      $|\ (X \neq a) \Rightarrow$ *syntax error* ***fi***

  fi

until *(Stack* ***empty****);* ***if*** $(y = \varepsilon) \Rightarrow$ *O.K.* $|\ (y \neq \varepsilon) \Rightarrow$ *syntax error* ***fi***

Recursive descent parser

function *pE:* ***if*** *(i=‘a’)* ***or*** *(i=‘(’) ⇒ pT; pE’*
| ***otherwise*** ⇒ *syntax error* ***fi***

function *pE’:* ***if*** *(i=‘+’)* ⇒ *verify(‘+’); pT; pE’*
| *(i=‘)’)* ***or*** *(i=‘ε’) ⇒ skip*
| ***otherwise*** ⇒ *syntax error* ***fi***

function *pT:* ***if*** *(i=‘a’)* ***or*** *(i=‘(’) ⇒ pF; pT’*
| ***otherwise*** ⇒ *syntax error* ***fi***

function *pT’:* ***if*** *(i=‘*’)* ⇒ *verify(‘*’); pF; pT’*
| *(i=‘+’)* ***or*** *(i=‘)’)* ***or*** *(i=‘ε’) ⇒ skip*
| ***otherwise*** ⇒ *syntax error* ***fi***

function *pF:* ***if*** *(i=‘a’)* ⇒ *verify(‘a’)*
| *(i=‘(’)* ⇒ *verify(‘(’); pE; verify(‘)’)*
| ***otherwise*** ⇒ *syntax error* ***fi***

i := 1:x; y := x:|x|-1; pE; ***if*** *(y = ε) ⇒ O.K. | (y ≠ ε) ⇒ syntax error* ***fi***