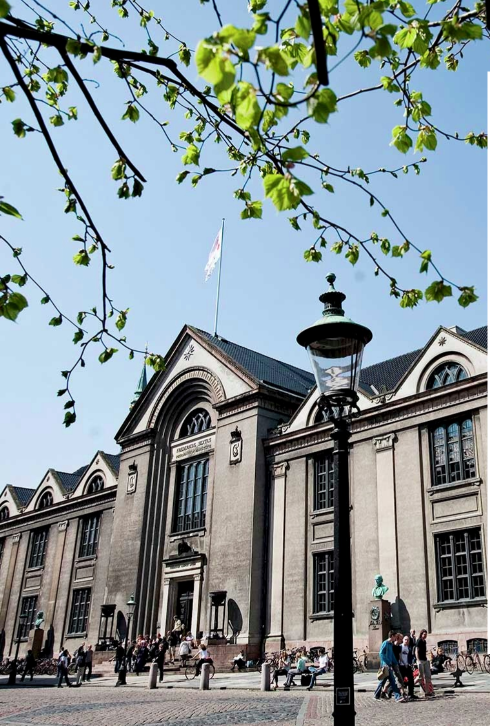

Parallel Parsing

The Implementation of a Parallel LL Parser Generator

William Henrich Due

30rd June 2023

KØBENHAVNS UNIVERSITET

What did I do?

LL parsing

Here is a LL(1) grammar.

$$1)\ T \rightarrow R \qquad 2)\ T \rightarrow aTc \qquad 3)\ R \rightarrow \varepsilon \qquad 4)\ R \rightarrow bR$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$(abc, T, ())$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$(abc, T, ()) \vdash (abc, aTc, 2)$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$(abc, T, ()) \vdash (abc, aTc, 2) \vdash (bc, Tc, 2)$$

LL parsing

Here is a LL(1) grammar.

$$1)\ \ T \to R \qquad 2)\ \ T \to aTc \qquad 3)\ \ R \to \varepsilon \qquad 4)\ \ R \to bR$$

$$(abc, T, ()) \vdash (abc, aTc, 2) \vdash (bc, Tc, 2) \vdash (bc, Rc, (2, 1))$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$\begin{aligned}(abc, T, ()) &\vdash (abc, aTc, 2) \vdash (bc, Tc, 2) \vdash (bc, Rc, (2,1)) \\ &\vdash (bc, bRc, (2,1,4))\end{aligned}$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$(abc, T, ()) \vdash (abc, aTc, 2) \vdash (bc, Tc, 2) \vdash (bc, Rc, (2,1)) \\ \vdash (bc, bRc, (2,1,4)) \vdash (c, Rc, (2,1,4))$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$\begin{aligned}(abc, T, ()) &\vdash (abc, aTc, 2) \vdash (bc, Tc, 2) \vdash (bc, Rc, (2,1)) \\ &\vdash (bc, bRc, (2,1,4)) \vdash (c, Rc, (2,1,4)) \\ &\vdash (c, c, (2,1,4,3))\end{aligned}$$

LL parsing

Here is a LL(1) grammar.

$$1)\ T \to R \qquad 2)\ T \to aTc \qquad 3)\ R \to \varepsilon \qquad 4)\ R \to bR$$

$$\begin{aligned}(abc, T, ()) &\vdash (abc, aTc, 2) \vdash (bc, Tc, 2) \vdash (bc, Rc, (2,1)) \\ &\vdash (bc, bRc, (2,1,4)) \vdash (c, Rc, (2,1,4)) \\ &\vdash (c, c, (2,1,4,3)) \vdash (\varepsilon, \varepsilon, (2,1,4,3))\end{aligned}$$

Accepted! the string "*abc*" can be parsed.

LLP Parsing

Augment the grammar, this grammar is LL(1) and LLP(1, 1).

$$0)\quad T' \to \vdash T \dashv$$

Now we parse the string "$\vdash abc \dashv$" instead.

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$

LLP Parsing

Augment the grammar, this grammar is LL(1) and LLP(1, 1).

$$0)\ \ T' \to \vdash T \dashv$$

Now we parse the string "$\vdash abc \dashv$" instead.

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$

1. Initial pushdown store.
2. Final pushdown store.
3. Left parse.

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$
\begin{tikzcd}[column sep=tiny]
 & & & (T', \varepsilon, (0,2,1,4,3)) \arrow[dl, dash] \arrow[dr, dash] & \\
 & & (T', \dashv, (0,2,1,4,3)) \arrow[dl, dash] \arrow[dr, dash] & & (\dashv, \varepsilon, ()) \\
 & (T', T c \dashv, (0,2)) \arrow[dl, dash] \arrow[d, dash] & & (Tc, \varepsilon, (1,4,3)) \arrow[d, dash] \arrow[dr, dash] & \\
(T', T \dashv, 0) & (T, Tc, 2) & & (T, R, (1,4)) & (Rc, \varepsilon, 3)
\end{tikzcd}
$$

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$(T', \varepsilon, (0,2,1,4,3))$$

$$(T', \dashv, (0,2,1,4,3)) \qquad (\dashv, \varepsilon, ())$$

$$(T', Tc\dashv, (0,2)) \qquad (Tc, \varepsilon, (1,4,3))$$

$$(T', T\dashv, 0) \quad (T, Tc, 2) \quad (T, R, (1,4)) \quad (Rc, \varepsilon, 3)$$

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$(T', \varepsilon, (0,2,1,4,3))$$

$$(T', \dashv, (0,2,1,4,3)) \qquad (\dashv, \varepsilon, ())$$

$$(T', Tc\dashv, (0,2)) \qquad (Tc, \varepsilon, (1,4,3))$$

$$(T', T\dashv, 0) \quad (T, Tc, 2) \quad (T, R, (1,4)) \quad (Rc, \varepsilon, 3)$$

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$(T', \varepsilon, (0, 2, 1, 4, 3))$$

$$(T', \dashv, (0, 2, 1, 4, 3)) \qquad (\dashv, \varepsilon, ())$$

$$(T', Tc\dashv, (0, 2)) \qquad (Tc, \varepsilon, (1, 4, 3))$$

$$(T', T\dashv, 0) \quad (T, Tc, 2) \quad (T, R, (1, 4)) \quad (Rc, \varepsilon, 3)$$

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$(T', \varepsilon, (0,2,1,4,3))$$

$$(T', \dashv, (0,2,1,4,3)) \qquad (\dashv, \varepsilon, ())$$

$$(T', Tc\dashv, (0,2)) \qquad (Tc, \varepsilon, (1,4,3))$$

$$(T', T\dashv, 0) \quad (T, Tc, 2) \quad (T, R, (1,4)) \quad (Rc, \varepsilon, 3)$$

LLP Parsing using Parallel Reduce

Use the associative **glue** operation.

$$(T', \varepsilon, (0,2,1,4,3))$$

$$(T', \dashv, (0,2,1,4,3)) \qquad (\dashv, \varepsilon, ())$$

$$(T', Tc\dashv, (0,2)) \qquad (Tc, \varepsilon, (1,4,3))$$

$$(T', T\dashv, 0) \quad (T, Tc, 2) \quad (T, R, (1,4)) \quad (Rc, \varepsilon, 3)$$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T\dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T\dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$
$(T', T\dashv)$	(T, Tc)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$
$(T', T \dashv)$	(T, Tc)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(T', \dashv T)$	(T, cT)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$
$(T', T \dashv)$	(T, Tc)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(T', \dashv T)$	(T, cT)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(\varepsilon, [^{\dashv}[^{T})$	$(]^{T}, [^{c}[^{T})$	$(]^{T}, [^{R})$	$(]^{R}]^{c}, \varepsilon)$	$(]^{\dashv}, \varepsilon)$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1,4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$
$(T', T \dashv)$	(T, Tc)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(T', \dashv T)$	(T, cT)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(\varepsilon, [^{\dashv}[^{T})$	$(]^{T}, [^{c}[^{T})$	$(]^{T}, [^{R})$	$(]^{R}]^{c}, \varepsilon)$	$(]^{\dashv}, \varepsilon)$

Now perform bracket matching and assert their types match up.

$$[^{\dashv}[^{T}]^{T}[^{c}[^{T}]^{T}[^{R}]^{R}]^{c}]^{\dashv}$$

LLP Parsing using Bracket Matching

$(\varepsilon, \vdash)$	$(\vdash, a)$	(a, b)	(b, c)	$(c, \dashv)$
$(T', T \dashv, 0)$	$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Rc, \varepsilon, 3)$	$(\dashv, \varepsilon, ())$
$(T', T \dashv)$	(T, Tc)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(T', \dashv T)$	(T, cT)	(T, R)	(Rc, ε)	$(\dashv, \varepsilon)$
$(\varepsilon, [^{\dashv}[^{T})$	$(]^{T}, [^{c}[^{T})$	$(]^{T}, [^{R})$	$(]^{R}]^{c}, \varepsilon)$	$(]^{\dashv}, \varepsilon)$

Now perform bracket matching and assert their types match up.

$$[^{\dashv}[^{T}]^{T}[^{c}[^{T}]^{T}[^{R}]^{R}]^{c}]^{\dashv}$$

Construct the production sequence.

$$0, 2, 1, 4, 3$$

The Problem

To perform LLP parsing a LLP table is needed.

	$\vdash$	a	b	c	$\dashv$
ε	$(T', T \dashv, 0)$				
$\vdash$		$(T, Tc, 2)$	$(T, R, (1, 4))$		$(T \dashv, \varepsilon, (1, 3))$
a		$(T, Tc, 2)$	$(T, R, (1, 4))$	$(Tc, \varepsilon, (1, 3))$	
b			$(R, R, 4)$	$(Rc, \varepsilon, 3)$	$(R \dashv, \varepsilon, 3)$
c				$(c, \varepsilon, ())$	$(\dashv, \varepsilon, ())$

1. Find the initial pushdown store for a specific symbol for the given lookahead and lookback.
2. Compute final pushdown store by LL parsing the first symbol of the lookahead using the initial pushdown store.
3. Construct LLP configuration using the initial, final pushdown store and left parse.

The Problem

Consider the following augmented LL(2) grammar.

$$0)\ S' \to \vdash S \dashv \qquad 1)\ A \to \varepsilon \qquad 2)\ S \to aAa \qquad 3)\ A \to a$$

The Problem

Consider the following augmented $\mathrm{LL}(2)$ grammar.

$$0)\ S' \to \vdash S \dashv \qquad 1)\ A \to \varepsilon \qquad 2)\ S \to aAa \qquad 3)\ A \to a$$

The PSLS (Prefix of a Suffix of a Leftmost Sentential) table. The grammar is LLP(2,2) since all the sets are singletons.

	$\dashv$	$a \dashv$	aa
$\vdash$			$\{S\}$
$\vdash a$		$\{A\}$	$\{A\}$
aa	$\{\dashv\}$	$\{a\}$	

The Problem

Consider the following augmented LL(2) grammar.

$$0)\ S' \rightarrow \vdash S \dashv \qquad 1)\ A \rightarrow \varepsilon \qquad 2)\ S \rightarrow aAa \qquad 3)\ A \rightarrow a$$

The PSLS (Prefix of a Suffix of a Leftmost Sentential) table. The grammar is LLP(2,2) since all the sets are singletons.

	$\dashv$	$a \dashv$	aa
$\vdash$			$\{S\}$
$\vdash a$		$\{A\}$	$\{A\}$
aa	$\{\dashv\}$	$\{a\}$	

A small example of a PSLS value.

$$S \Rightarrow_{lm}^{*} \vdash S \dashv \Rightarrow \vdash aAa \dashv \Rightarrow^{*} \vdash aa \dashv$$

This corresponds to the entry PSLS$(\vdash a, a \dashv) = \{A\}$.

The Problem

Construct the LL(2) table.

	$\vdash a$	aa	$a \dashv$
S'	$S' \to \vdash S \dashv$		
S		$S \to aAa$	
A		$A \to a$	$A \to \varepsilon$

The Problem

Construct the LL(2) table.

	$\vdash a$	aa	$a \dashv$
S'	$S' \rightarrow \vdash S \dashv$		
S		$S \rightarrow aAa$	
A		$A \rightarrow a$	$A \rightarrow \varepsilon$

Try LL parsing the initial pushdown store $\text{PSLS}(\vdash a, a \dashv) = \{A\}$.

$$(a \dashv, A, ()) \vdash (a \dashv, \varepsilon, 1)$$

Due to this the final pushdown store cannot be determined since the first symbol can not be parsed.

The Problem

Construct the LL(2) table.

	$\vdash a$	aa	$a \dashv$
S'	$S' \to \vdash S \dashv$		
S		$S \to aAa$	
A		$A \to a$	$A \to \varepsilon$

The Problem

Construct the LL(2) table.

	$\vdash a$	aa	$a \dashv$
S'	$S' \to \vdash S \dashv$		
S		$S \to aAa$	
A		$A \to a$	$A \to \varepsilon$

Try LL parsing the initial pushdown store $\text{PSLS}(\vdash a, a \dashv) = \{A\}$.

$$(a \dashv, A, ()) \vdash (a \dashv, \varepsilon, 1)$$

Due to this the final pushdown store cannot be determined since the first symbol can not be parsed.

The Problem

The problem is due to the PSLS definition.

$$\begin{aligned}\mathrm{PSLS}(x, y) = \{&\alpha : \exists S \Rightarrow_{lm}^{*} wuA\beta \Rightarrow wxB\gamma \Rightarrow^{*} wxy\delta,\\ &w, u \in T^{*}, A, B \in N, \alpha, \beta, \gamma, \delta \in (N \cup T)^{*}, u \neq x,\\ &\alpha \text{ is the shortest prefix of } B\gamma \text{ such that } \mathrm{FIRST}(y) \subseteq \mathrm{FIRST}_1(\alpha)\}\\ \cup \{&a : \exists S \Rightarrow^{*} wuA\beta \Rightarrow wxa\gamma \Rightarrow^{*} wxy\delta,\\ &a = \mathrm{FIRST}_1(y), w, u \in T^{*}, \beta, \gamma, \delta \in (N \cup T)^{*}, u \neq x\}\end{aligned}$$

The Problem

The problem is due to the PSLS definition.

$$\begin{aligned}\mathrm{PSLS}(x, y) = \{\alpha : \exists S \Rightarrow^*_{lm} wuA\beta \Rightarrow wxB\gamma \Rightarrow^* wxy\delta,\\ w, u \in T^*, A, B \in N, \alpha, \beta, \gamma, \delta \in (N \cup T)^*, u \neq x,\\ \alpha \text{ is the shortest prefix of } B\gamma \text{ such that } \mathrm{FIRST}(y) \subseteq \mathrm{FIRST}_1(\alpha)\}\\ \cup \{a : \exists S \Rightarrow^* wuA\beta \Rightarrow wxa\gamma \Rightarrow^* wxy\delta,\\ a = \mathrm{FIRST}_1(y), w, u \in T^*, \beta, \gamma, \delta \in (N \cup T)^*, u \neq x\}\end{aligned}$$

The problem is.

$$\mathrm{PSLS}(x, y) \Rightarrow a \text{ where } ab = y \text{ and } a \in T$$

does not imply

$$(y, \mathrm{PSLS}(x, y), ()) \vdash^* (b, \mathrm{PSLS}(x, y), \pi)$$

The Problem

The problem is due to the PSLS definition.

$$\begin{aligned}
\mathrm{PSLS}_k(x, y) = \{&\alpha : \exists S \Rightarrow^*_{lm} wuA\beta \Rightarrow wxB\gamma \Rightarrow^* wxy\delta, \\
&w, u \in T^*, A, B \in N, \alpha, \beta, \gamma, \delta \in (N \cup T)^*, u \neq x, \\
&\alpha \text{ is the shortest prefix of } B\gamma \text{ such that } y \in \mathrm{FIRST}_k(\alpha)\} \\
\cup \{&y : \exists S \Rightarrow^* wuA\beta \Rightarrow wxa\gamma \Rightarrow^* wxy\delta, \\
&a = \mathrm{FIRST}_1(y), w, u \in T^*, \beta, \gamma, \delta \in (N \cup T)^*, u \neq x\}
\end{aligned}$$

The Problem

Construct the new PSLS_2 table.

	$\dashv$	$a\dashv$	aa
$\vdash$			$\{S\}$
$\vdash a$		$\{Aa\dashv\}$	$\{Aa\}$
aa	$\{\dashv\}$	$\{a\dashv\}$	

The Problem

Construct the new PSLS_2 table.

	$\dashv$	$a \dashv$	aa
$\vdash$			$\{S\}$
$\vdash a$		$\{Aa \dashv\}$	$\{Aa\}$
aa	$\{\dashv\}$	$\{a \dashv\}$	

Try LL parsing the initial pushdown store $\text{PSLS}_2(\vdash a, a \dashv) = \{A\}$.

$$(a \dashv, Aa \dashv, ()) \vdash (a \dashv, a \dashv, 1) \vdash (\dashv, \dashv, 1)$$

Now the final pushdown store can be found.

Applications