# Ph.D. defense

Petr Šaloun

December 14, 2001



## Ph.D. defense outline

1. Title
2. Outline
3. Sequential $LR$ parsing
4. Parallel parsing architecture
5. Parallel $LRP(1,1)$ parsing
6. $LRP(q,k)$ grammars and languages
7. Conclusions.

**Example 1**

Let us consider the augmented strong $LR(1)$ grammar

$$G_1 = (\{S, E, E', T, T', F\}, \{a, +, *, (, )\}, P, S)$$

with rules:

0. $S \to E$
1. $E \to E'T$
2. $E' \to \varepsilon$
3. $E' \to E+$
4. $F \to (E)$
5. $F \to a$
6. $T \to T'F$
7. $T' \to \varepsilon$
8. $T' \to T*$



The parsing table for the (strong) grammar $G_1$:

| | ( | ) | $*$ | $+$ | $a$ | $\varepsilon$ |
|---|---|---|---|---|---|---|
| $S$ | | | | | | ACC |
| $E$ | | SHF | | SHF | | $R_0$ |
| $E'$ | $R_7$ | | | | $R_7$ | |
| $F$ | | $R_6$ | $R_6$ | $R_6$ | | $R_6$ |
| $T$ | | $R_1$ | $S$ | $R_1$ | | $R_1$ |
| $T'$ | SHF | | | | SHF | |
| ( | $R_2$ | | | | $R_2$ | |
| ) | | $R_4$ | $R_4$ | $R_4$ | | $R_4$ |
| $*$ | $R_8$ | | | | $R_8$ | |
| $+$ | $R_3$ | | | | $R_3$ | |
| $a$ | | $R_5$ | $R_5$ | $R_5$ | | $R_5$ |
| # | $R_2$ | | | | $R_2$ | |

- $R_i$ means the REDUCE action by rule *i*,
- SHF means the SHIFT action,
- ACC means the ACCEPT action,
- empty entry means an ERROR action.

The example of sequential strong $LR$ parsing of $a * a + a$ by $G_1$ :

$$
\begin{array}{llll}
 & (\# & , a * a + a, \varepsilon) & \\
\vdash & (\#E' & , a * a + a, 2) & \\
\vdash & (\#E'T' & , a * a + a, 27) & \textsc{Shf}(a) \\
\vdash & (\#E'T'a & , \; *a + a, 27) & \\
\hline
\vdash & (\#E'T'F & , \; *a + a, 275) & \\
\vdash & (\#E'T & , \; *a + a, 2756) & \textsc{Shf}(*) \\
\vdash & (\#E'T* & , \; a + a, 2756) & \\
\hline
\vdash & (\#E'T' & , \; a + a, 27568) & \textsc{Shf}(a) \\
\vdash & (\#E'T'a & , \; +a, 27568) & \\
\hline
\vdash & (\#E'T'F & , \; +a, 275685) & \\
\vdash & (\#E'T & , \; +a, 2756856) & \\
\vdash & (\#E & , \; +a, 27568561) & \textsc{Shf}(+) \\
\vdash & (\#E+ & , \; a, 27568561) & \\
\hline
\vdash & (\#E' & , \; a, 275685613) & \\
\vdash & (\#E'T' & , \; a, 2756856137) & \textsc{Shf}(a) \\
\vdash & (\#E'T'a & , \; \varepsilon, 2756856137) & \\
\vdash & (\#E'T'F & , \; \varepsilon, 27568561375) & \\
\vdash & (\#E'T & , \; \varepsilon, 275685613756) & \\
\vdash & (\#E & , \; \varepsilon, 2756856137561) & \\
\vdash & (\#S & , \; \varepsilon, 27568561375610) & \\
\end{array}
$$



**Conclusion (sequential)**

The parsing process: the continuous sequences of REDUCE actions finished by SHIFT actions.

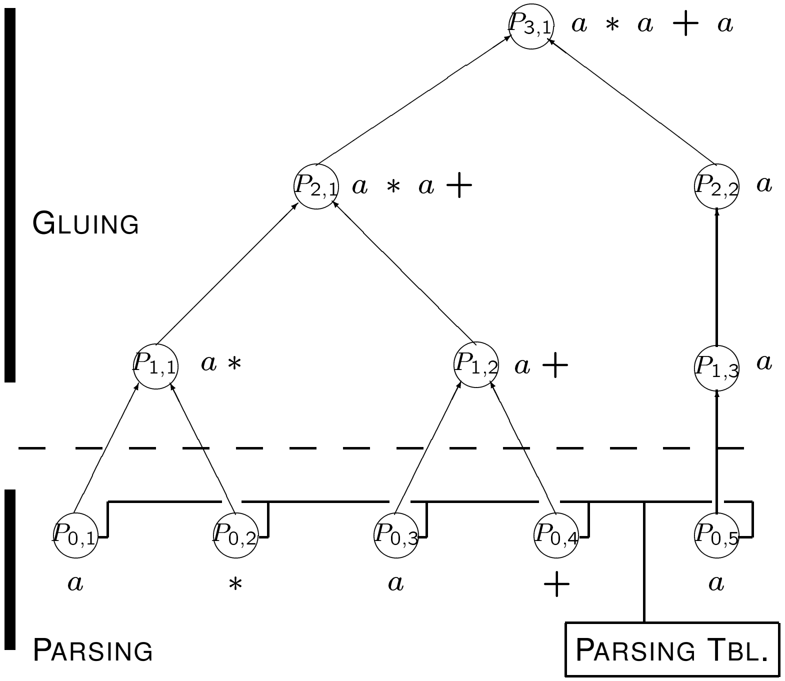
**Idea (parallel)**

The subprocess finishes its work by the SHIFT action for its last input symbol.

One subprocess does partial parsing of a substring of the input string. The substring of length one (terminal) symbol is the finest granularity.

The knowledge of the *left adjacent input* symbol (terminal) defines the top symbol of the pushdown store at the start of the partial parsing (sub)process in the parallel case.

Architecture for the parallel parsing,
two *phases* of parallel parsing: *parsing* and *gluing*.




The input string is $a * a + a$.



The algorithm suitable for parallel gluing (*binary associative* operation) is called *parallel binary reduction*.

The most natural interconnection network is a complete binary tree.

The parallel reduction is a scalable parallel algorithm.

## Example 2

$LRP(1,1)$ augmented grammar

$G_2 = (\{S, E, E', T, T', F\}, \{a, +, *, (, ), \vdash, \dashv\}, P, S)$, with rules $P$:

0. $S \to \vdash E \dashv$
1. $E \to E'T$
2. $E' \to \varepsilon$
3. $E' \to E+$
4. $F \to (E)$
5. $F \to a$
6. $T \to T'F$
7. $T' \to \varepsilon$
8. $T' \to T*$



Parsing table for the $LRP(1,1)$ grammar $G_2$

| | ( | ) | $*$ | $+$ | $a$ | $\dashv$ |
|---|---|---|---|---|---|---|
| $S$ | | | | | | ACC |
| $E$ | | SHF | | SHF | | $R_0$ |
| $E'$ | $R_7$ | | | | $R_7$ | |
| $F$ | | $R_6$ | $R_6$ | $R_6$ | | $R_6$ |
| $T$ | | $R_1$ | $S$ | $R_1$ | | $R_1$ |
| $T'$ | SHF | | | | SHF | |
| ( | $R_2$ | | | | $R_2$ | |
| ) | | $R_4$ | $R_4$ | $R_4$ | | $R_4$ |
| $*$ | $R_8$ | | | | $R_8$ | |
| $+$ | $R_3$ | | | | $R_3$ | |
| $a$ | | $R_5$ | $R_5$ | $R_5$ | | $R_5$ |
| $\vdash$ | $R_2$ | | | | $R_2$ | |

Computing of $(\alpha, \omega, \pi)$ triples for (some) admissible pairs of strings $(u, v)$. $LRP(1,1)$ grammar $G_2$.

$$
\begin{array}{lll}
 & (\vdash, a, \varepsilon) & \alpha = \vdash \\
(\vdash, a) & \vdash (\vdash E', a, 2) & \\
 & \vdash (\vdash E'T', a, 27) & \\
 & \vdash (\vdash E'T'a, \varepsilon, 27) & (\vdash, \vdash E'T'a, 27) \\
\hline
 & (a, +, \varepsilon) & \alpha = a \\
 & \vdash (F, +, 5) & \\
(a, +) & \vdash (T, +, 56) & \alpha = T'a \\
 & \vdash (E, +, 561) & \alpha = E'T'a \\
 & \vdash (E+, \varepsilon, 561) & (E'T'a, E+, 561) \\
 & (a, \dashv, \varepsilon) & \alpha = a \\
 & \vdash (F, \dashv, 5) & \\
(a, \dashv) & \vdash (T, \dashv, 56) & \alpha = T'a \\
 & \vdash (E, \dashv, 561) & \alpha = E'T'a \\
 & \vdash (S, \varepsilon, 5610) & (\vdash E'T'a, S, 5610) \\
\hline
 & (+, a, \varepsilon) & \alpha = + \\
(+, a) & \vdash (E', a, 3) & \alpha = E+ \\
 & \vdash (E'T', a, 37) & \\
 & \vdash (E'T'a, \varepsilon, 37) & (E+, E'T'a, 37)
\end{array}
$$



Admissible pairs of strings for $LRP(1,1)$ grammar $G_2$

| $u$ | $v$ |
|---|---|
| $\vdash$ | $a, ($ |
| $a$ | $+, *, ), \dashv$ |
| $+$ | $a, ($ |
| $*$ | $a, ($ |
| $)$ | $+, *, ), \dashv$ |
| $($ | $a, ($ |

...strong parsing table.

| | $a$ | $+$ | $*$ | $($ | $)$ | $\dashv$ |
|---|---|---|---|---|---|---|
| $\vdash$ | 27 | | | 27 | | |
| $a$ | | 561 | 56 | | 561 | 5610 |
| $+$ | 37 | | | 37 | | |
| $*$ | 8 | | | 8 | | |
| $)$ | | 461 | 46 | | 461 | 4610 |
| $($ | 27 | | | 27 | | |

Grammar $G_2$,
$LRP(1,1)$ parsing table.

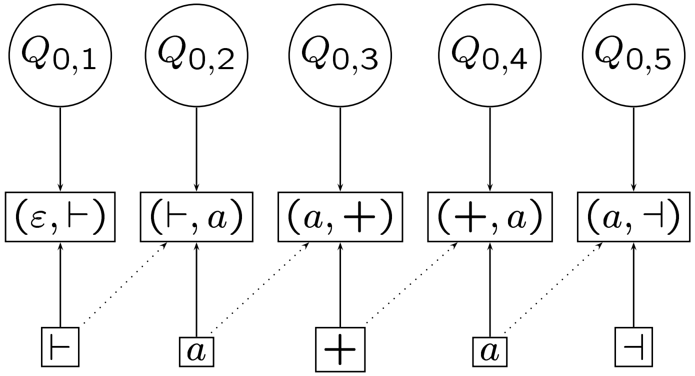
| | $a$ | $+$ | $*$ |
|---|---|---|---|
| $\vdash$ | $(\vdash, \vdash E'T'a, 27)$ | | |
| $a$ | | $(E'T'a, E+, 561)$ | $(T'a, T*, 56)$ |
| $+$ | $(E+, E'T'a, 37)$ | | |
| $*$ | $(T*, T'a, 8)$ | | |
| $)$ | | $(E'T'(E), T+, 461)$ | $(T'(E), T*, 46)$ |
| $($ | $((, (E'T'a, 27)$ | | |

| | $($ | $)$ | $\dashv$ |
|---|---|---|---|
| $\vdash$ | $(\vdash, \vdash E'T'(, 27)$ | | |
| $a$ | | $(E'T'a, E), 561)$ | $(\vdash E'T'a, S, 5610)$ |
| $+$ | $(E+, E'T'(, 37)$ | | |
| $*$ | $(T*, T'(, 8)$ | | |
| $)$ | | $(E'T'(E), T), 461)$ | $(E'T'(E), S, 4610)$ |
| $($ | $((, (E'T'(, 27)$ | | |



Grammar $G_2$,
$LRP(1,1)$ parallel parsing of $\vdash a + a \dashv$

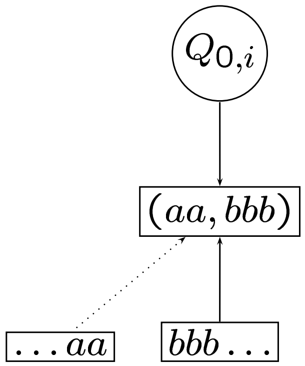
Marking of input processors.



| $\vdash$ | $a$ | $+$ | $a$ | $\dashv$ |
|---|---|---|---|---|
| $Q_{0,1}(\varepsilon, \vdash)$ | $Q_{0,2}(\vdash, a)$ | $Q_{0,3}(a, +)$ | $Q_{0,4}(+, a)$ | $Q_{0,5}(a, \dashv)$ |
| $(\varepsilon, \vdash, \varepsilon)$ | $(\vdash, \vdash E'T'a, 27)$ | $(E'T'a, E+, 561)$ | $(E+, E'T'a, 37)$ | $(\vdash E'T'a, S, 5610)$ |
| $Q_{1,1}(\varepsilon, \vdash a)$ | | $Q_{1,2}(a, +a)$ | | $Q_{1,3}(a, \dashv)$ |
| $(\varepsilon, \vdash, \varepsilon) \oplus (\vdash, \vdash E'T'a, 27)$<br>$(\varepsilon, \vdash E'T'a, 27)$ | | $(E'T'a, E+, 561) \oplus (E+, E'T'a, 37)$<br>$(E'T'a, E'T'a, 56137)$ | | $(\vdash E'T'a, S, 5610)$ |
| $Q_{2,1}(\varepsilon, \vdash a + a)$ | | | | $Q_{2,2}(a, \dashv)$ |
| $(\varepsilon, \vdash E'T'a, 27) \oplus (E'T'a, E'T'a, 56137)$<br>$(\varepsilon, \vdash E'T'a, 2756137)$ | | | | $(\vdash E'T'a, S, 5610)$ |
| $Q_{3,1}(\varepsilon, \vdash a + a \dashv)$ | | | | |
| $(\varepsilon, \vdash E'T'a, 2756137) \oplus (\vdash E'T'a, S, 5610)$<br>$(\varepsilon, S, 27561375610)$ | | | | |



# From $LRP(1,1)$ to $LRP(q,k)$ languages




$LRP(2,3)$

## Informal summary

**Limited locality** *admissible pair* $(u, v)$, *lookback string* $u$ of the length $q$, and the *lookahead string* $v$ of the length $k$.

**Limited history** String naming a set of items reflects the history.

$LRP(q, k)$ **item** extends the $LR(k)$ item by the lookback string and the lookahead string.

$l - strong\ LRP(q, k)$ **grammar** uses names of sets of $LRP(q, k)$ items of length $l$.

$LRP$ **grammars** are proper subset of $LR$ grammars.



## $LRP(q, k)$ grammars

$G_3 = (\{a, b, \vdash, \dashv\}, \{S', S, A, B\}, P, S')$

0. $S' \to \vdash S \dashv$
1. $S \to AB$
2. $A \to a$
3. $B \to Bb$
4. $B \to \varepsilon$

The $G_3$ grammar is $LRP(2, 1)$ grammar,
but it is not $LRP(1, 2)$ grammar.

$$
\begin{aligned}
b = \{\ & [A \to ab\centerdot C, \dashv, b, b \dashv] \\
& [C \to \centerdot b, \dashv, b, b \dashv] \\
& [B \to b\centerdot, \dashv, b, \dashv] \\
& [C \to b\centerdot, \dashv, b, \dashv]\}
\end{aligned}
$$

If $G$, and $G'$, are $LRP(q, k)$, and $LRP(l, m)$,
and let $q, k, l, m \in \mathbb{N}$ and $q + k = l + m$,
then it is **not true** that the languages are equal, i.e.

$$L(G) \neq L(G')$$

## $LRP(q,k)$ grammars and regular grammars

Let consider augmented grammar
$G_4 = (\{S', S\}, \{\vdash, \dashv, a\}, P, S')$, with rules $P$:

0. $S' \to \vdash S \dashv$
1. $S \to aaS$
2. $S \to \varepsilon$

$L(G_4) = \{a^{2n} : n > 0\}$.

The middle processor does not know whether the $a$ is about to a process even or odd one for any $q, k$.

The computation of triples $(\alpha, \omega, \pi)$ fails.

$$
(aa, \dashv) \quad
\begin{array}{ll}
(aa, \dashv, \varepsilon) & \alpha = aa \\
\vdash (aaS, \dashv, 2) & \\
\vdash (S, \dashv, 21) & \alpha = aa \\
\text{ERROR} &
\end{array}
$$



## Conclusions

- new ideas for parallel parsing based on strong $LR$ grammars;
- $LRP(q,k)$ new subclass of context–free $LR(k)$ grammars;
- new nontrivial algorithms for deterministic parallel parsing.