On LR Parsing with Selective Delays

E. Bertsch | M.-J. Nederhof | S. Schmitz

Ruhr-U. Bochum | U. St Andrews | ENS Cachan

CC 2013, March 22nd

Rome, Italy

Selective ML

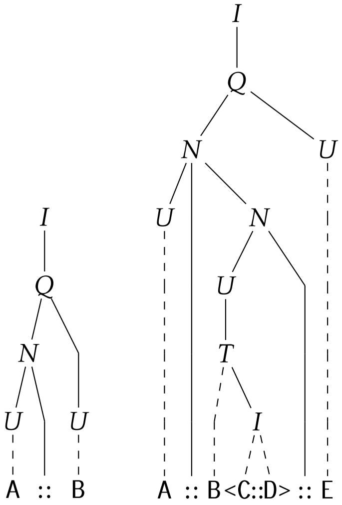
named after Marcus (1980) and Leermakers (1992)

C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$
\begin{aligned}
I &\to U \mid Q \\
U &\to i \mid T \\
Q &\to N\,U \\
N &\to U :: N \mid U :: \\
T &\to i<I>
\end{aligned}
$$



C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

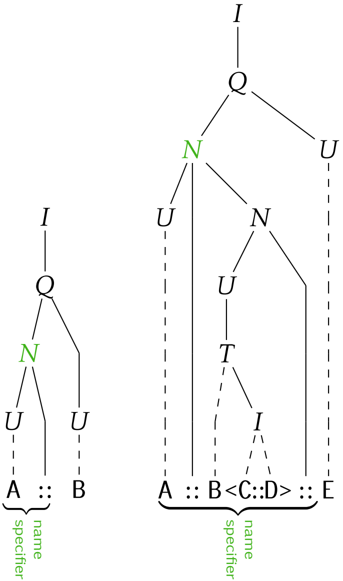
$$I \rightarrow U \mid Q$$

$$U \rightarrow i \mid T$$

$$Q \rightarrow N\,U$$

$$N \rightarrow U \,::\, N \mid U \,::$$

$$T \rightarrow i<I>$$

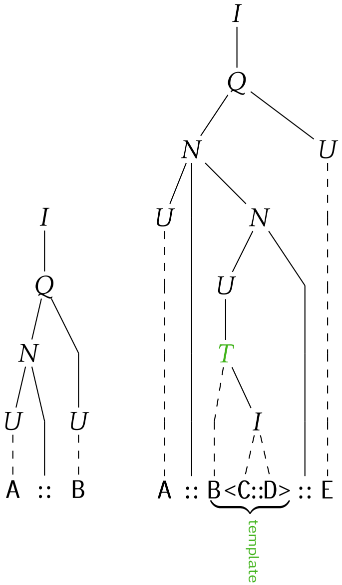


C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$I \to U \mid Q$$
$$U \to i \mid T$$
$$Q \to N\,U$$
$$N \to U :: N \mid U ::$$
$$T \to i \mathord{<} I \mathord{>}$$



C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$I \to U \mid Q$$
$$U \to i \mid T$$
$$Q \to N\,U$$
$$N \to U :: N \mid U ::$$
$$T \to i <I>$$

GNU/Bison output:

- `conflicts: 1 shift/reduce`
- `warning: rule useless in parser due to conflicts:`

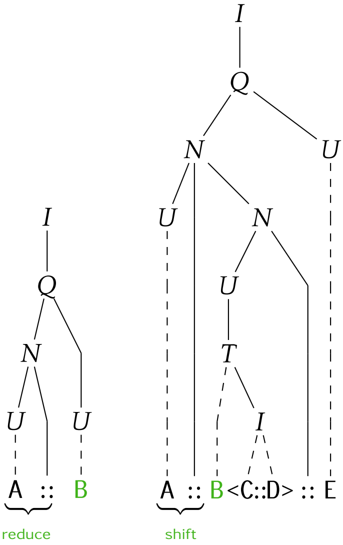
$$N \to U ::$$

C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$I \rightarrow U \mid Q$$
$$U \rightarrow i \mid T$$
$$Q \rightarrow N\,U$$
$$N \rightarrow U :: N \mid U ::$$
$$T \rightarrow i <I>$$



C++ Qualified Identifiers

(ISO Standard, 1998)

Solutions?

Grammar:

$$I \to U \mid Q$$
$$U \to i \mid T$$
$$Q \to N\,U$$
$$N \to U :: N \mid U ::$$
$$T \to i \texttt{<} I \texttt{>}$$

- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$I \to U \mid Q$$
$$U \to i \mid T$$
$$Q \to N\,U$$
$$N \to U :: N \mid U ::$$
$$T \to i \texttt{<} I \texttt{>}$$

Solutions?

- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$
\begin{aligned}
I &\rightarrow U \mid Q \\
U &\rightarrow i \mid T \\
Q &\rightarrow N\,U \\
N &\rightarrow U :: N \mid U :: \\
T &\rightarrow i < I >
\end{aligned}
$$

Solutions?

- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

Grammar:

$$I \rightarrow U \mid Q$$
$$U \rightarrow i \mid T$$
$$Q \rightarrow N\,U$$
$$N \rightarrow U :: N \mid U ::$$
$$T \rightarrow i \texttt{<} I \texttt{>}$$

Solutions?

- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

left-recursive Grammar:

$$I \rightarrow U \mid Q$$
$$U \rightarrow i \mid T$$
$$Q \rightarrow N\,U$$
$$N \rightarrow N\,U :: \mid U ::$$
$$T \rightarrow i <I>$$

Solutions?

- priority mechanisms

  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

combed Grammar:

$$I \rightarrow U \mid Q$$
$$U \rightarrow i \mid T$$
$$Q \rightarrow [NU]$$
$$N \rightarrow U :: N \mid U ::$$
$$T \rightarrow i<I>$$

Solutions?

- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

combed Grammar:

$$I \rightarrow U \mid Q$$
$$U \rightarrow i \mid T$$
$$Q \rightarrow [NU]$$
$$[NU] \rightarrow U :: [NU] \mid U :: U$$
$$T \rightarrow i \lt I \gt$$

Solutions?

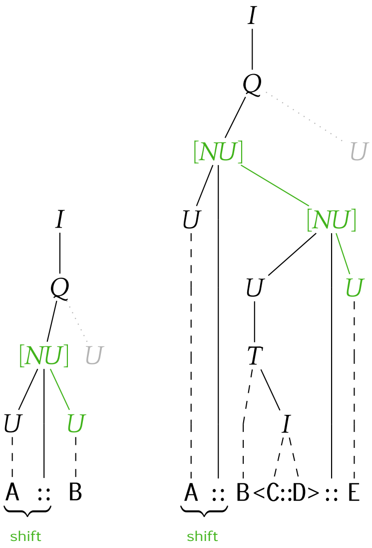
- priority mechanisms
  (Bouwers et al., 2008)
- GLR
- refactor the grammar
  - language-preserving
  - large space of possible transformations
    (see e.g. Lämmel, 2001)

C++ Qualified Identifiers

(ISO Standard, 1998)

combed Grammar:

$$
\begin{aligned}
I &\rightarrow U \mid Q \\
U &\rightarrow i \mid T \\
Q &\rightarrow [NU] \\
[NU] &\rightarrow U :: [NU] \mid U :: U \\
T &\rightarrow i \lt I \gt
\end{aligned}
$$



Going Formal: Combings

Write a context-free grammar as $G = \langle N, \Sigma, P, S \rangle$, with vocabulary $V = \Sigma \uplus N$.

$G \hookrightarrow G'$ if $\exists \mu : V'^* \to V^*$ a homomorphism s.t.

1. $\mu(S') = S$,
2. $\forall a \in \Sigma, \mu(a) = a$,
3. $\mu(N') \subseteq N \cdot V^*$
4. $\{A' \to \mu(\alpha') \mid A' \to \alpha' \in P'\}$
   $= \{A' \to \alpha\delta \mid A' \in N', \mu(A') = A\delta$, and $A \to \alpha \in P\}$

We call it a k-combing ($\hookrightarrow_k$) if $\mu(N') \subseteq N \cdot V^{\leqslant k}$.

Going Formal: Combings

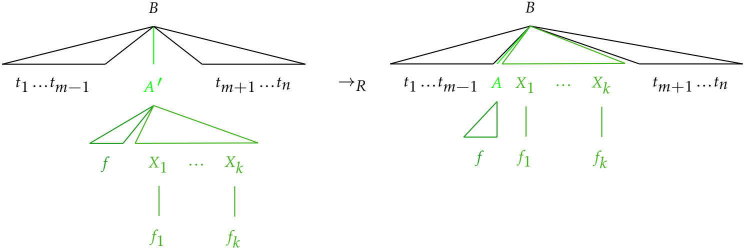
Example

$$
\begin{array}{rl@{\qquad}rl}
I & \to U \mid Q & I & \to U \mid Q \\
U & \to i \mid T & U & \to i \mid T \\
Q & \to NU & Q & \to [NU] \\
N & \to U :: N \mid U :: & [NU] & \to U :: [NU] \mid U :: U \\
T & \to i < I > & T & \to i < I >
\end{array}
$$

Define $\mu([NU]) = N\,U$ and $\mu(A) = A$ for the other nonterminals A.

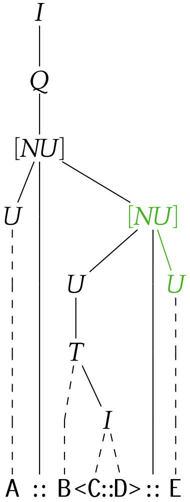
(Un-)Combings as Tree Mappings

One rule per nonterminal A' s.t. $\mu(A') = A X_1 \cdots X_q$:



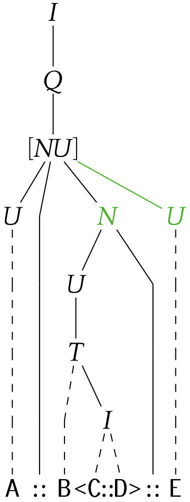
(Un-)Combings as Tree Mappings

Example



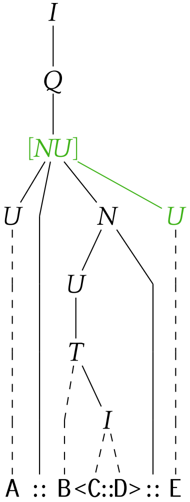
(Un-)Combings as Tree Mappings

Example



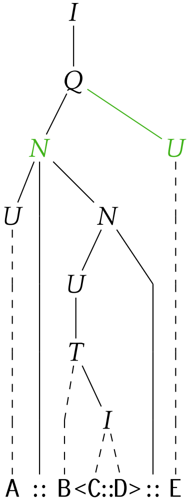
(Un-)Combings as Tree Mappings

Example



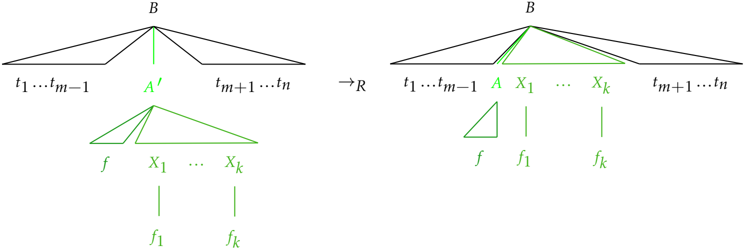
(Un-)Combings as Tree Mappings

Example



(Un-)Combings as Tree Mappings

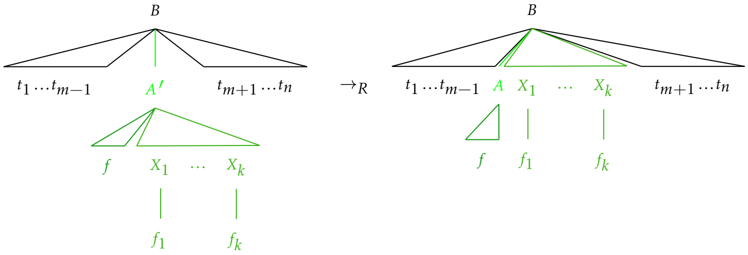
One rule per nonterminal A' s.t.
$\mu(A') = AX_1 \cdots X_q$:



Formally, a forest rewriting system, which is confluent, terminating, and preserves terminal yields:

(Un-)Combings as Tree Mappings

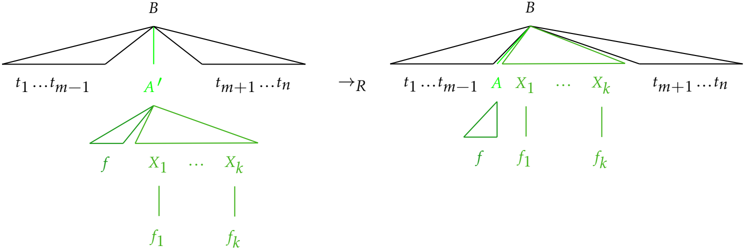
One rule per nonterminal A' s.t.
$\mu(A') = AX_1 \cdots X_q$:



Formally, a forest rewriting system, which is confluent, terminating, and preserves terminal yields:

(Un-)Combings as Tree Mappings

One rule per nonterminal A' s.t. $\mu(A') = AX_1 \cdots X_q$:



Formally, a forest rewriting system, which is confluent, terminating, and preserves terminal yields:

Theorem

If $G \hookrightarrow G'$, *then* $L(G) = L(G')$.

Selective ML

Definition

G is selML(k, m) if $\exists G', G\#^k \hookrightarrow_k G'$ and G' is LR(m).

Combing Strategies

Given k:

- always comb "as much context as possible"?
  ML(k,m) grammars (Bertsch and Nederhof, 2007)
- always comb "as little as possible"?

Selective ML

Definition

G is selML(k, m) if $\exists G', G\#^k \hookrightarrow_k G'$ and G' is LR(m).

Combing Strategies

Given k:

- always comb "as much context as possible"?
  ML(k,m) grammars (Bertsch and Nederhof, 2007)
- always comb "as little as possible"?

Selective ML

Definition

G is selML(k, m) if $\exists G', G\#^k \hookrightarrow_k G'$ and G' is LR(m).

Combing Strategies

Given k:

- always comb "as much context as possible"?
  ML(k,m) grammars (Bertsch and Nederhof, 2007)
- always comb "as little as possible"?

Selective ML

Definition

G is selML(k, m) if $\exists G', G\#^k \hookrightarrow_k G'$ and G' is LR(m).

Combing Strategies

Given k:

- always comb "as much context as possible"? ML(k,m) grammars (Bertsch and Nederhof, 2007)
- always comb "as little as possible"?

SelML(k,m) parsers

Principles

- LR-like construction
- add context only when required
- recompute states in case of conflicts: distinguished conflict and deprecated items.
- a combing can be extracted from the parser

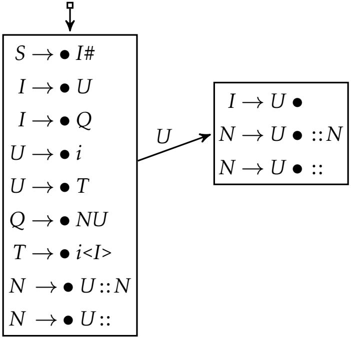
SelML(k,m) parsers

Example

$S \to \bullet\ I\#$
$I \to \bullet\ U$
$I \to \bullet\ Q$
$U \to \bullet\ i$
$U \to \bullet\ T$
$Q \to \bullet\ NU$
$T \to \bullet\ i{<}I{>}$
$N \to \bullet\ U::N$
$N \to \bullet\ U::$

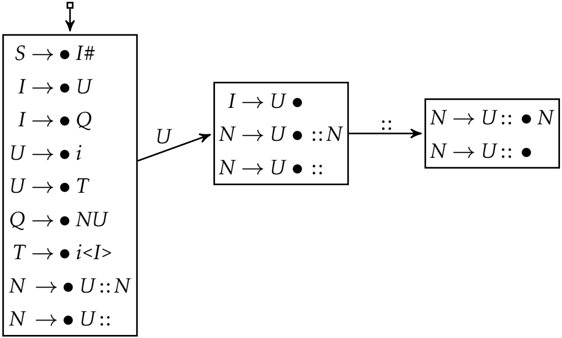
SELML(k,m) PARSERS

EXAMPLE



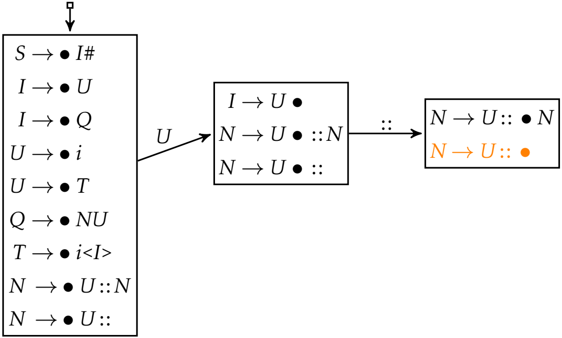
SelML(k,m) parsers

Example



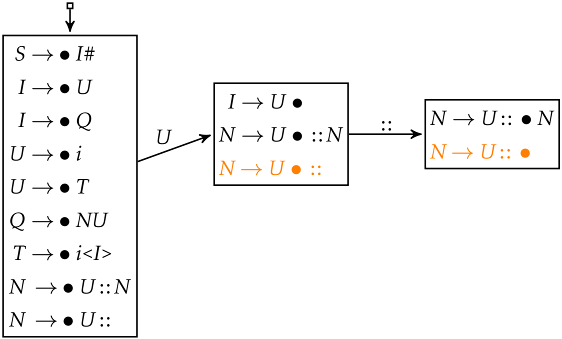
SelML(k,m) parsers

Example



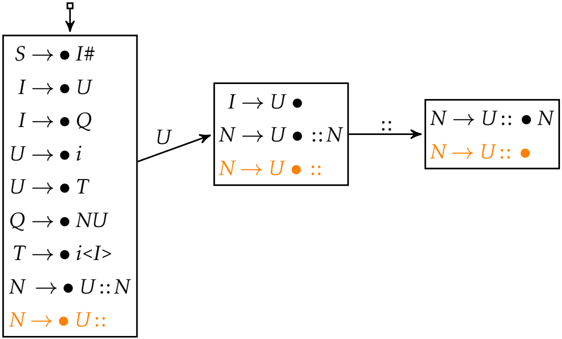
SelML(k,m) parsers

Example



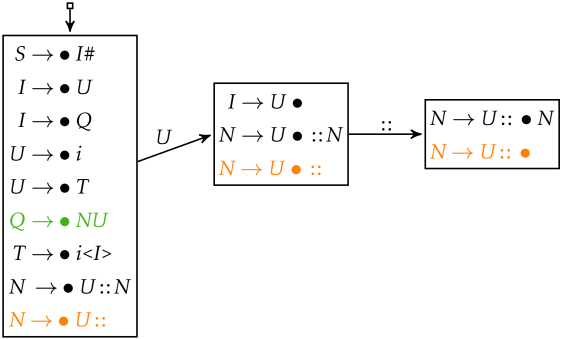
SelML(k,m) parsers

Example



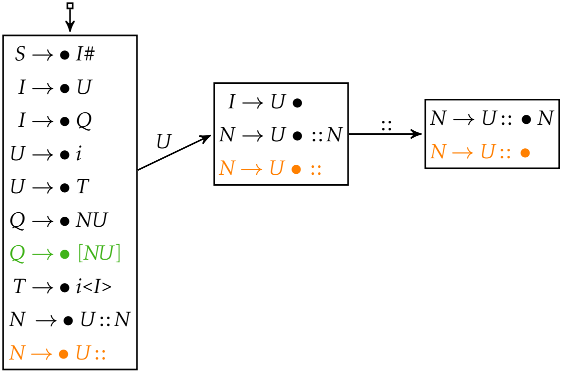
SelML(k,m) parsers

Example



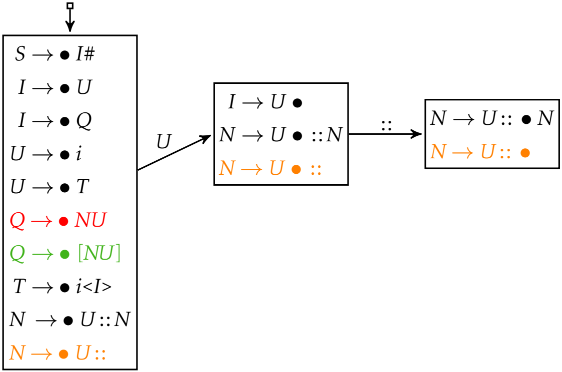
SelML(k,m) parsers

Example



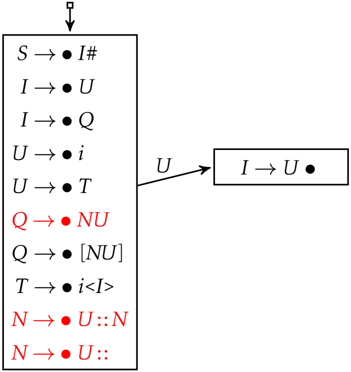
SelML(k,m) parsers

Example



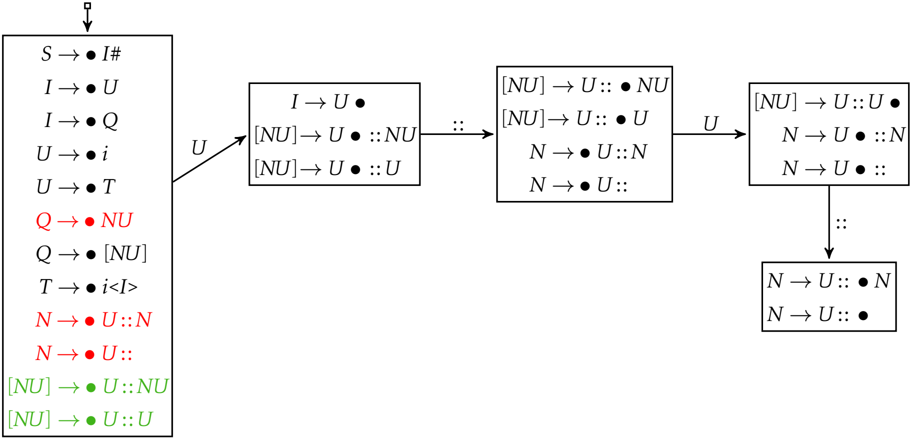
SelML(k,m) parsers

Example



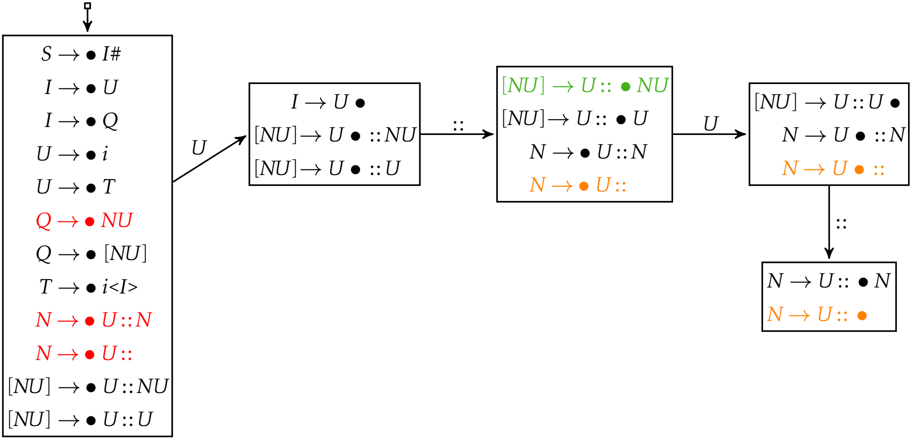
SelML(k,m) parsers

Example



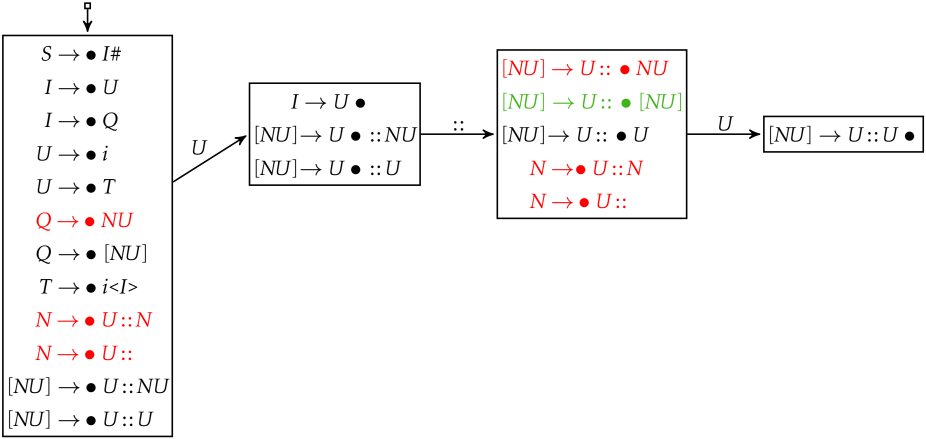
SelML(k,m) parsers

Example



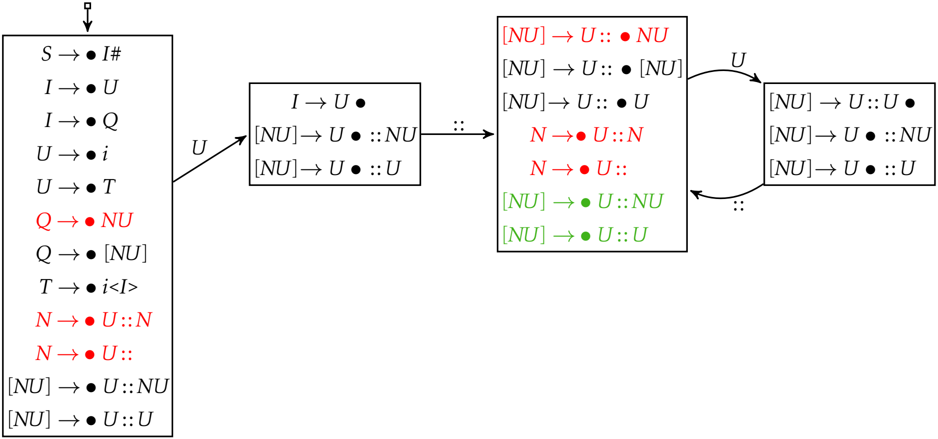
SelML(k,m) parsers

Example



SelML(k,m) parsers

Example



Theoretical Results

For each k and m:

grammar class selML(k,m) includes ML(k,m), and thus LR(m)

monotonicity selML(k,m) implies selML(k',m') for all $k' \geqslant k$ and $m' \geqslant m$

unambiguity of selML grammars

deterministic the language of a selML grammar is a DCFL

undecidability of whether a grammar is selML(k,m) for some k and m

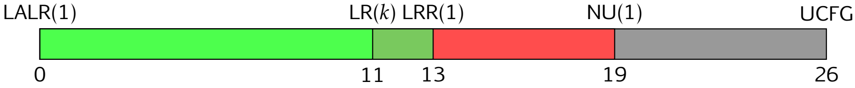
Experimental Results

- proof of concept implementation available at http://www.cs.st-andrews.ac.uk/~mjn/code/mlparsing/
- grammar classes on 26 small grammars (Basten, 2008; Schmitz, 2010), $k + m \leqslant 50$:

- parser sizes:
  - often much smaller than ML
  - inconclusive wrt. LR

Experimental Results

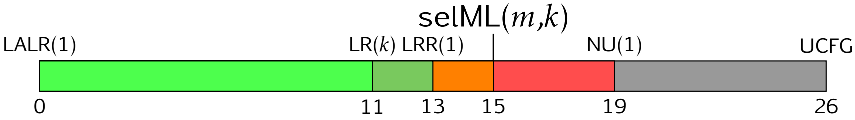
- proof of concept implementation available at http://www.cs.st-andrews.ac.uk/~mjn/code/mlparsing/
- grammar classes on 26 small grammars (Basten, 2008; Schmitz, 2010), $k + m \leqslant 50$:



- parser sizes:
  - often much smaller than ML
  - inconclusive wrt. LR

Experimental Results

- proof of concept implementation available at
  http://www.cs.st-andrews.ac.uk/~mjn/code/mlparsing/
- grammar classes on 26 small grammars (Basten, 2008; Schmitz, 2010), $k+m \leqslant 50$:

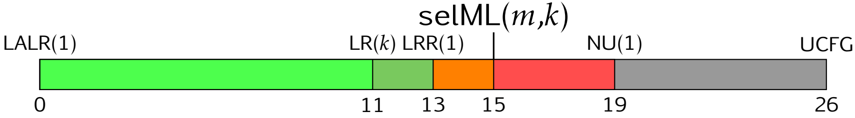


- parser sizes:
  - often much smaller than ML
  - inconclusive wrt. LR

Experimental Results

- proof of concept implementation available at
  http://www.cs.st-andrews.ac.uk/~mjn/code/mlparsing/

- grammar classes on 26 small grammars (Basten, 2008; Schmitz, 2010), $k + m \leqslant 50$:



- parser sizes:
  - often much smaller than ML
  - inconclusive wrt. LR

Concluding Remarks

- avoid (some) LR conflicts
- deterministic parsing: linear-time complexity and exclusion of ambiguities
- grammar transformation view
- LR-like parser: can be used for generalized parsing.

References

Basten, H.J.S., 2008. The usability of ambiguity detection methods for context-free grammars. In Vinju, J. and Johnstone, A., editors, *LDTA'08*, volume 238(5) of *Elec. Notes in Theor. Comput. Sci.*, pages 35–46. Elsevier. doi:10.1016/j.entcs.2009.09.039.

Bertsch, E. and Nederhof, M.J., 2007. Some observations on LR-like parsing with delayed reduction. *Information Processing Letters*, 104(6):195–199. doi:10.1016/j.ipl.2007.07.003.

Bouwers, E., Bravenboer, M., and Visser, E., 2008. Grammar engineering support for precedence rule recovery and compatibility checking. In *LDTA 2007*, volume 203(2) of *Elec. Notes in Theor. Comput. Sci.*, pages 85–101. Elsevier. doi:10.1016/j.entcs.2008.03.046.

ISO Standard, 1998. *ISO/IEC 14882:1998: Programming Languages — C++*. International Organization for Standardization, Geneva, Switzerland.

Lämmel, R., 2001. Grammar adaptation. In Oliveira, J.N. and Zave, P., editors, *FME 2001*, volume 2021 of *LNCS*, pages 550–570. Springer. doi:10.1007/3-540-45251-6_32.

Leermakers, R., 1992. Recursive ascent parsing: from Earley to Marcus. *Theor. Comput. Sci.*, 104(2):299–312. doi:10.1016/0304-3975(92)90127-2.

Marcus, M.P., 1980. *A Theory of Syntactic Recognition for Natural Language*. Series in Artificial Intelligence. MIT Press.

Schmitz, S., 2010. An experimental ambiguity detection tool. *Science of Computer Programming*, 75(1–2):71–84. doi:10.1016/j.scico.2009.07.002.