Verifying Modular Grammars

Sylvain Schmitz

LORIA, INRIA Nancy - Grand Est, Nancy, France

Systèmes à Événements Discrets, LIAFA
April 28, 2008

Contents

Grammar Engineering

Klint et al. [2005]

- grammars as specifications
  - rich toolsets (parsers, pretty printers, etc.)
- grammars as programs
  - methodology
  - testing
  - verification

Grammar Engineering

Klint et al. [2005]

- grammars as specifications
  - rich toolsets (parsers, pretty printers, etc.)
- grammars as programs
  - methodology
  - testing
  - verification

Case Study: Modular Syntax

Modules

- meaningful subsets
- reusable
- composable

Comparing parsers

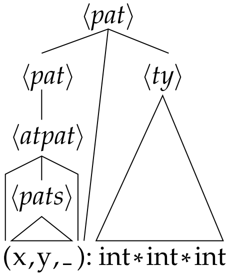
- two grammar fragments $\mathcal{G}_1$ and $\mathcal{G}_2$
- composition $\mathcal{G}_1 \cup \mathcal{G}_2$ using
  - LR parsers
  - GLR parsers
  - parsing expression grammars

SML Pattern Syntax

Standard ML [Milner et al., 1997]

$$
\begin{array}{rcl}
\langle pat \rangle & \rightarrow & \langle atpat \rangle \mid \langle pat \rangle : \langle ty \rangle \\
\langle atpat \rangle & \rightarrow & vid \mid \_ \mid (\ \langle pats \rangle \ ) \mid (\ ) \\
\langle pats \rangle & \rightarrow & \langle pat \rangle \mid \langle pats \rangle \ , \ \langle pat \rangle
\end{array}
\qquad (\mathcal{G}_1)
$$

Example

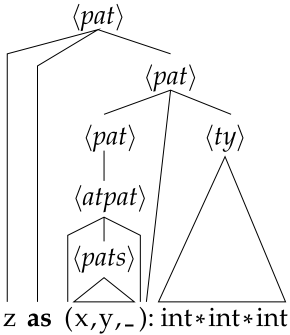


SML Layered Pattern Syntax

Standard ML [Milner et al., 1997]

$$\langle pat \rangle \quad \rightarrow \quad vid : \langle ty \rangle \; as \; \langle pat \rangle \,|\, vid \; as \; \langle pat \rangle \qquad (\mathcal{G}_2)$$

Example



LR Parsers

Crespi Reghizzi and Psaila [1998]

- parser for a LR(k) grammar, works in $\mathcal{O}(n)$
- but DCFL not closed under union:

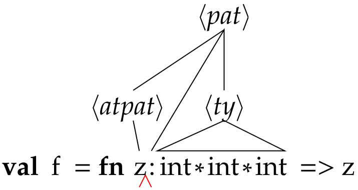
val f = **fn** z : int ∗ int ∗ int

- $\mathcal{G}_1 \cup \mathcal{G}_2$ is not LR(k) for any k

LR Parsers

Crespi Reghizzi and Psaila [1998]

- parser for a LR(k) grammar, works in $\mathcal{O}(n)$
- but DCFL not closed under union:

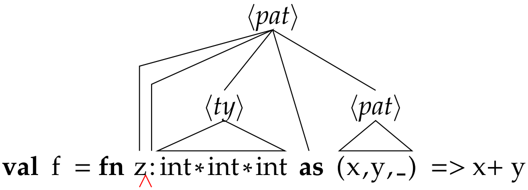


- $\mathcal{G}_1 \cup \mathcal{G}_2$ is not LR(k) for any k

LR Parsers

Crespi Reghizzi and Psaila [1998]

- parser for a LR(k) grammar, works in $\mathcal{O}(n)$
- but DCFL not closed under union:



- $\mathcal{G}_1 \cup \mathcal{G}_2$ is not LR(k) for any k

LR Parsers

Crespi Reghizzi and Psaila [1998]

- parser for a LR(k) grammar, works in $\mathcal{O}(n)$
- but DCFL not closed under union:

val f = **fn** z : int∗int∗int∗int∗int ∗...

- $\mathcal{G}_1 \cup \mathcal{G}_2$ is not LR(k) for any k

Generalized LR Parsers

e.g. SDF2 [Visser, 1997]

- parser for any CFG, works in $\mathcal{O}(2^{|G|} n^3)$
- CFL closed under union

```
module G1
exports
  sorts PAT ATPAT
  context-free syntax
    ATPAT                  -> PAT
    PAT ":" TY             -> PAT
    "(" {PAT ","}* ")"     -> ATPAT
    "_"                    -> ATPAT
    VID                    -> ATPAT
```

```
module G1UG2
imports
  G1
exports
  context-free syntax
    VID (":" TY)? "as" PAT -> PAT
```

- but UCFL isn't!

Generalized LR Parsers

e.g. SDF2 [Visser, 1997]

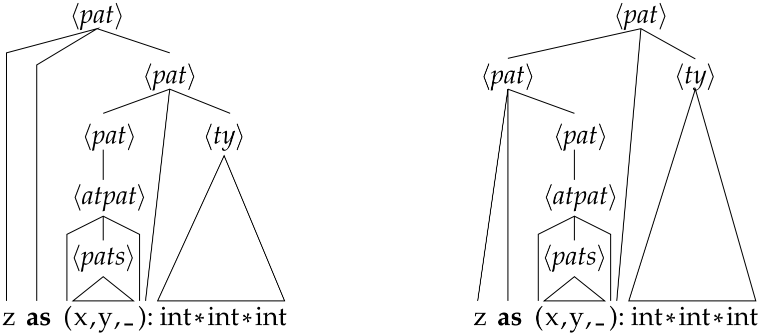
- parser for any CFG, works in $\mathcal{O}(2^{|G|}\, n^3)$
- CFL closed under union

```
module G1
exports
  sorts PAT ATPAT
  context-free syntax
    ATPAT                   -> PAT
    PAT ":" TY              -> PAT
    "(" {PAT ","}* ")"      -> ATPAT
    "_"                     -> ATPAT
    VID                     -> ATPAT
```

```
module G1UG2
imports
  G1
exports
  context-free syntax
    VID (":" TY)? "as" PAT -> PAT
```

- but UCFL isn't!

Ambiguity



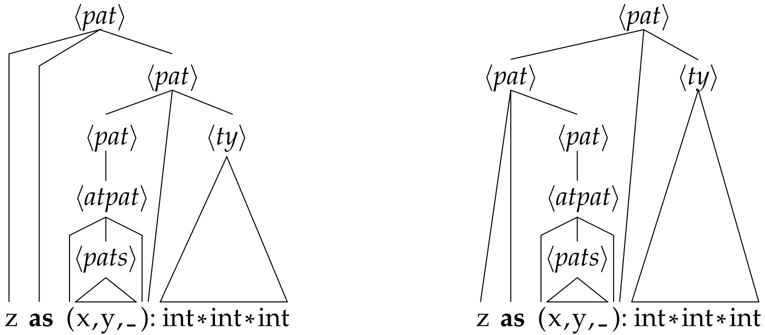
- run-time error:

```
sglr:error: Ambiguity in input, line 1, col 0:
  PAT  ":"  TY -> PAT;VID  (":" TY )?  "as"  PAT -> PAT
```

- choose between alternative parses:

```
VID (":" TY)? "as" PAT      -> PAT {prefer}
```

Ambiguity



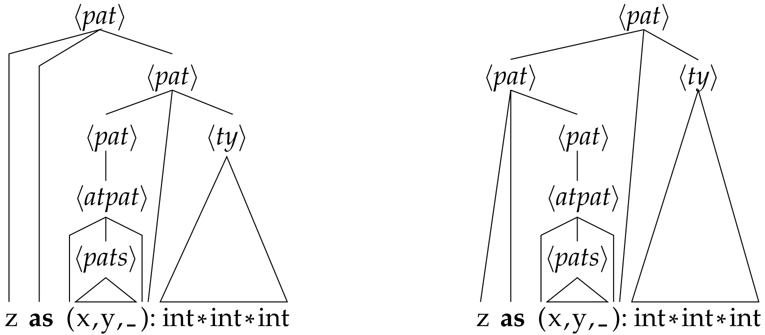
- run-time error:

```
sglr:error: Ambiguity in input, line 1, col 0:
  PAT  ":"  TY -> PAT;VID  (":" TY )?  "as"  PAT -> PAT
```

- choose between alternative parses:

```
VID (":" TY)? "as" PAT      -> PAT {prefer}
```

Ambiguity



- run-time error:

```
sglr:error: Ambiguity in input, line 1, col 0:
  PAT  ":"  TY -> PAT;VID  (":" TY )?  "as"  PAT -> PAT
```

- choose between alternative parses:

```
    VID (":" TY)? "as" PAT      -> PAT {prefer}
```

Parsing Expression Grammars

Birman and Ullman [1973], Ford [2004]

- ordered rules $A \leftarrow \alpha_1/\alpha_2/\ldots/\alpha_n$
- $\mathcal{O}(|G|\,n)$ recursive backtracking top-down parsing
- closed under union, intersection, complement
- includes DCFL

Example

- $A \leftarrow aa/a$: $\mathcal{L}(A) = \{a, aa\}$
- $A \leftarrow a/aa$: $\mathcal{L}(A) = \{a\}$
- $A \leftarrow Ba, B \leftarrow aa/a$: $\mathcal{L}(A) = \{aaa\}$
- $A \leftarrow aAa/aa$: $\mathcal{L}(A) = \{a^{2^n} \mid n \geqslant 1\}$

Parsing Expression Grammars

Birman and Ullman [1973], Ford [2004]

- ordered rules $A \leftarrow \alpha_1/\alpha_2/\ldots/\alpha_n$
- $\mathcal{O}(|G|\, n)$ recursive backtracking top-down parsing
- closed under union, intersection, complement
- includes DCFL

Example

- $A \leftarrow aa/a$: $\mathcal{L}(A) = \{a, aa\}$
- $A \leftarrow a/aa$: $\mathcal{L}(A) = \{a\}$
- $A \leftarrow Ba, B \leftarrow aa/a$: $\mathcal{L}(A) = \{aaa\}$
- $A \leftarrow aAa/aa$: $\mathcal{L}(A) = \{a^{2^n} \mid n \geqslant 1\}$

Parsing Expression Grammars

Birman and Ullman [1973], Ford [2004]

- ordered rules $A \leftarrow \alpha_1/\alpha_2/\ldots/\alpha_n$
- $\mathcal{O}(|G|\,n)$ recursive backtracking top-down parsing
- closed under union, intersection, complement
- includes DCFL

Example

- $A \leftarrow aa/a$: $\mathcal{L}(A) = \{a, aa\}$
- $A \leftarrow a/aa$: $\mathcal{L}(A) = \{a\}$
- $A \leftarrow Ba, B \leftarrow aa/a$: $\mathcal{L}(A) = \{aaa\}$
- $A \leftarrow aAa/aa$: $\mathcal{L}(A) = \{a^{2^n} \mid n \geqslant 1\}$

Parsing Expression Grammars

Birman and Ullman [1973], Ford [2004]

- ordered rules $A \leftarrow \alpha_1/\alpha_2/\dots/\alpha_n$
- $\mathcal{O}(|G|\,n)$ recursive backtracking top-down parsing
- closed under union, intersection, complement
- includes DCFL

Example

- $A \leftarrow aa/a$: $\mathcal{L}(A) = \{a, aa\}$
- $A \leftarrow a/aa$: $\mathcal{L}(A) = \{a\}$
- $A \leftarrow Ba, B \leftarrow aa/a$: $\mathcal{L}(A) = \{aaa\}$
- $A \leftarrow aAa/aa$: $\mathcal{L}(A) = \{a^{2^n} \mid n \geqslant 1\}$

Parsing Expression Grammars

Birman and Ullman [1973], Ford [2004]

- ordered rules $A \leftarrow \alpha_1/\alpha_2/\ldots/\alpha_n$
- $\mathcal{O}(|G|\,n)$ recursive backtracking top-down parsing
- closed under union, intersection, complement
- includes DCFL

Example

- $A \leftarrow aa/a$: $\mathcal{L}(A) = \{a, aa\}$
- $A \leftarrow a/aa$: $\mathcal{L}(A) = \{a\}$
- $A \leftarrow Ba$, $B \leftarrow aa/a$: $\mathcal{L}(A) = \{aaa\}$
- $A \leftarrow aAa/aa$: $\mathcal{L}(A) = \{a^{2^n} \mid n \geqslant 1\}$

Packrat Parsers

e.g. Rats! [Grimm, 2006]

```
module G1;
public generic Pattern =
    <Atomic>   AtomicPattern  TypeOp ;
generic TypeOp  = (void:":":Symbol Type )? ;
generic AtomicPattern =
    <Tuple>    void:"(":Symbol PatternList? void:")":Symbol
  / <Wildcard> "_":Symbol
  / <Variable> ValueID ;
generic PatternList = Pattern ( void:",":Symbol Pattern )* ;

module G1UG2;
 modify G1;
generic Pattern  +=
    <Atomic> ...
  / <Layered>  ValueID TypeOp void:"as":Keyword Pattern TypeOp ;
```

But... Disjointness

- run-time error:

```
xtc.parser.ParseException:
input:1:2: error: symbol characters expected
z as (x,y,_):int*int*int
  ^
```

- order the rules differently:

```
generic Pattern  +=
    <Layered>  ValueID TypeOp void:"as":Keyword Pattern TypeOp ;
  / <Atomic> ...
```

Summary

- tools for modular syntax
- undecidable issues
  - ambiguity in CFGs
  - disjointness in PEGs
- need for verification
- need for approximations

Bracketed Grammars

$$\mathcal{G} = \langle N, T, P, S \rangle,\ V = N \cup T$$

$$\begin{aligned}
\langle pat \rangle &\xrightarrow{1} \langle atpat \rangle \\
\langle pat \rangle &\xrightarrow{2} \langle pat \rangle\ :\ \langle ty \rangle \\
\langle atpat \rangle &\xrightarrow{3} vid \\
\langle atpat \rangle &\xrightarrow{4} _ \\
\langle atpat \rangle &\xrightarrow{5} (\ \langle pats \rangle\ ) \\
\langle atpat \rangle &\xrightarrow{6} (\ ) \\
\langle pats \rangle &\xrightarrow{7} \langle pat \rangle \\
\langle pats \rangle &\xrightarrow{8} \langle pats \rangle\ ,\ \langle pat \rangle \\
\langle pat \rangle &\xrightarrow{9} vid\ :\ \langle ty \rangle\ \mathsf{as}\ \langle pat \rangle \\
\langle pat \rangle &\xrightarrow{10} vid\ \mathsf{as}\ \langle pat \rangle
\end{aligned}$$

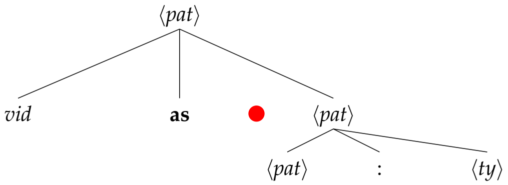
Bracketed Grammars

$$\mathcal{G}_b = \langle N, T_b, P_b, S\rangle,\ V_b = N \cup T_b$$

$$\begin{array}{rcl}
\langle pat\rangle & \xrightarrow{1} & d_1\ \langle atpat\rangle\ r_1 \\
\langle pat\rangle & \xrightarrow{2} & d_2\ \langle pat\rangle\ :\ \langle ty\rangle\ r_2 \\
\langle atpat\rangle & \xrightarrow{3} & d_3\ vid\ r_3 \\
\langle atpat\rangle & \xrightarrow{4} & d_4\ _\ r_4 \\
\langle atpat\rangle & \xrightarrow{5} & d_5\ (\ \langle pats\rangle\ )\ r_5 \\
\langle atpat\rangle & \xrightarrow{6} & d_6\ (\ )\ r_6 \\
\langle pats\rangle & \xrightarrow{7} & d_7\ \langle pat\rangle\ r_7 \\
\langle pats\rangle & \xrightarrow{8} & d_8\ \langle pats\rangle\ ,\ \langle pat\rangle\ r_8 \\
\langle pat\rangle & \xrightarrow{9} & d_9\ vid\ :\ \langle ty\rangle\ \text{as}\ \langle pat\rangle\ r_9 \\
\langle pat\rangle & \xrightarrow{10} & d_{10}\ vid\ \text{as}\ \langle pat\rangle\ r_{10}
\end{array}$$

Position Graph Γ

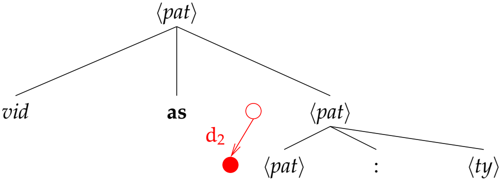
Left-to-right Walks in Trees



$$\mathrm{d}_{10}\ \textit{vid}\ \mathbf{as} \bullet \mathrm{d}_2\ \langle \textit{pat} \rangle\ :\ \langle \textit{ty} \rangle\ \mathrm{r}_2\ \mathrm{r}_{10}$$

Position Graph Γ

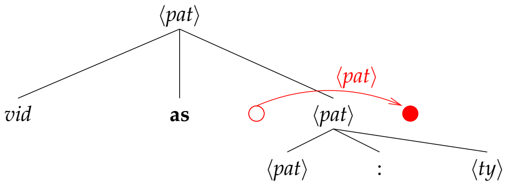
Left-to-right Walks in Trees



$$d_{10}\ \textit{vid}\ \textbf{as}\ d_2 \bullet \langle pat \rangle\ :\ \langle ty \rangle\ r_2\ r_{10}$$

Position Graph Γ

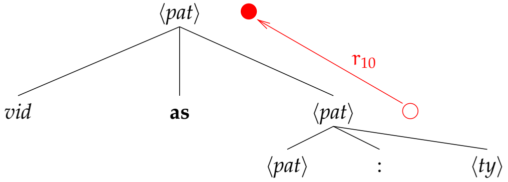
Left-to-right Walks in Trees



d_{10} *vid* **as** d_2 ⟨*pat*⟩ : ⟨*ty*⟩ r_2 • r_{10}

Position Graph Γ

Left-to-right Walks in Trees



$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle\ r_2\ r_{10}\ \bullet$$

Position Automaton $\Gamma/\equiv$

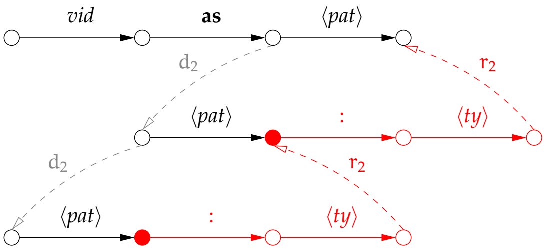
Definition

$\Gamma/\equiv$ is the quotient of Γ by an equivalence relation $\equiv$ between positions.

Language over-approximation

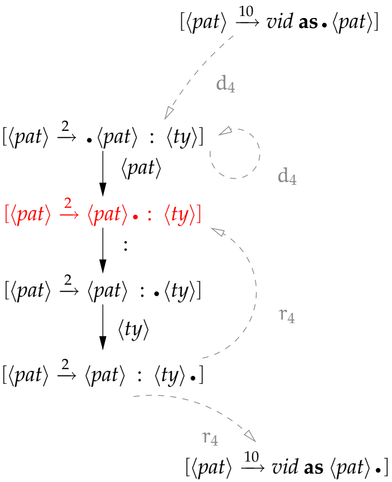
$$\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv) \cap T_b^*$$

Example: item_0 Equivalence



- equivalence class $[\langle pat\rangle \xrightarrow{2} \langle pat\rangle \bullet\ :\ \langle ty\rangle]$
- LR(0) items
- Γ/item_0: nondeterministic LR(0) automaton

Example: item$_0$ Equivalence

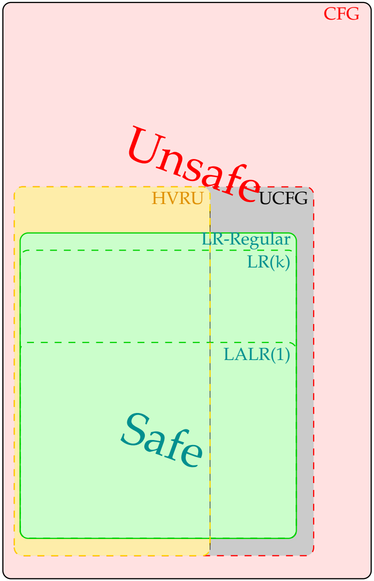


Summary

- framework for approximations
- applications:
  - parser construction
  - grammar verification

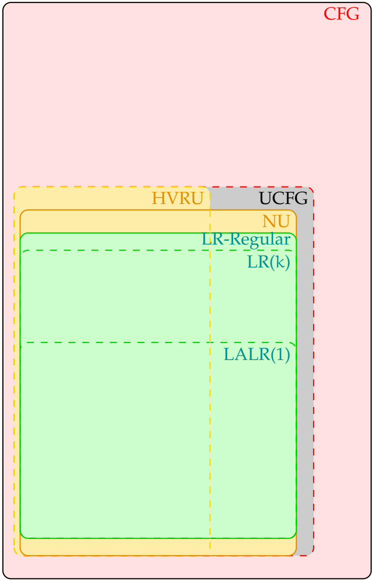
Unambiguous Grammar Classes

- LR(k) [Knuth, 1965]
- LR-Regular [Čulik and Cohen, 1973]
- Horizontal and vertical unambiguity test [Brabrand et al., 2007]
- Unambiguous CFGs [Cantor, 1962, Chomsky and Schützenberger, 1963]



Unambiguous Grammar Classes

- LR(k) [Knuth, 1965]
- LR-Regular [Čulik and Cohen, 1973]
- Horizontal and vertical unambiguity test [Brabrand et al., 2007]
- Unambiguous CFGs [Cantor, 1962, Chomsky and Schützenberger, 1963]



Principles

- a bracketed sentence = a derivation tree
- ambiguity = more than one tree with the same yield

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ :\ \langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2 d_{10}\ \mathit{vid}\ \mathbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

- construct a FSA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\mathcal{A})$, and look for bracketed sentences with the same yield

Principles

- a bracketed sentence = a derivation tree
- ambiguity = more than one tree with the same yield

$$\mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \mathrm{d}_2\ \mathrm{d}_1\ \mathrm{d}_3\ \mathit{vid}\ \mathrm{r}_3\ \mathrm{r}_1\ :\ \langle \mathit{ty} \rangle\ \mathrm{r}_2\ \mathrm{r}_{10}$$
$$\mathrm{d}_2\,\mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \mathrm{d}_1\ \mathrm{d}_3\ \mathit{vid}\ \mathrm{r}_3\ \mathrm{r}_1\ \mathrm{r}_{10}\ :\ \langle \mathit{ty} \rangle\ \mathrm{r}_2$$

- construct a FSA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\mathcal{A})$, and look for bracketed sentences with the same yield

Principles

- a bracketed sentence = a derivation tree
- ambiguity = more than one tree with the same yield

$$\mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \mathrm{d}_2\ \mathrm{d}_1\ \mathrm{d}_3\ \mathit{vid}\ \mathrm{r}_3\ \mathrm{r}_1\ :\ \langle \mathit{ty} \rangle\ \mathrm{r}_2\ \mathrm{r}_{10}$$

$$\mathrm{d}_2\,\mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \mathrm{d}_1\ \mathrm{d}_3\ \mathit{vid}\ \mathrm{r}_3\ \mathrm{r}_1\ \mathrm{r}_{10}\ :\ \langle \mathit{ty} \rangle\ \mathrm{r}_2$$

- construct a FSA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{G}_b) \subseteq \mathcal{L}(\mathcal{A})$, and look for bracketed sentences with the same yield

Regular Unambiguity

Definition

- h bracket erasing homomorphism
- $\mathcal{G}$ is *regular unambiguous* for $\equiv$ of *finite index*, if there does not exist $w_b \neq w'_b$ in $\mathcal{L}(\Gamma/\equiv) \cap \mathrm{T}_b^*$ with $\mathrm{h}(w_b) = \mathrm{h}(w'_b)$

Squaring Algorithms

- NFA ambiguity [Even, 1965]
- functionality [Béal et al., 2003]
- mutual accessibility relations between pairs of states

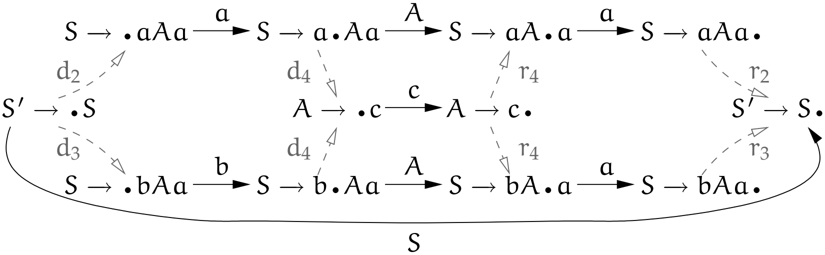
Horizontal and Vertical Ambiguity

Brabrand et al. [2007]

Definition

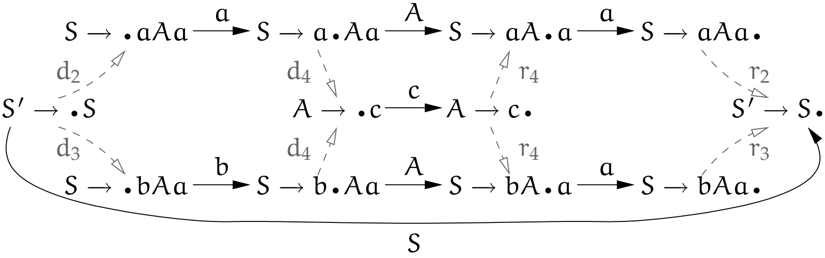
- $\mathcal{G}$ is *vertically unambiguous* iff $\forall A \to \alpha_1, A \to \alpha_2 \in P, \alpha_1 \neq \alpha_2, \mathcal{L}(\alpha_1) \cap \mathcal{L}(\alpha_2) = \emptyset$
- $\mathcal{G}$ is *horizontally unambiguous* iff $\forall A \to \alpha$ and $\forall \alpha_1, \alpha_2$ with $\alpha = \alpha_1 \alpha_2, \mathcal{L}(\alpha_1) \not\bowtie \mathcal{L}(\alpha_2) = \emptyset$
- $L_1 \not\bowtie L_2 = \{xyz \mid x, xy \in L_1, y \in T^+, \text{ and } yz, z \in L_2\}$
- $\mathcal{O}(|\mathcal{G}|^5)$ algorithm
- $\mathrm{RU}(\equiv) \subset \mathrm{HVRU}(\equiv)$

LR(k) condition



- $S \xrightarrow{2} aAa,\ S \xrightarrow{3} bAa,\ A \xrightarrow{4} c$
- $\mathrm{LR}(0) \not\subseteq \mathrm{RU}(\mathrm{item}_0)$
- regular approximations are too weak

LR(k) condition



- $S \xrightarrow{2} aAa$, $S \xrightarrow{3} bAa$, $A \xrightarrow{4} c$
- $\mathrm{LR}(0) \nsubseteq \mathrm{RU}(\mathrm{item}_0)$
- regular approximations are too weak

Nonterminal Transitions

- $\mathcal{SF}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv)$
- look for two different bracketed sentential forms in $\mathcal{L}(\Gamma/\equiv)$

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \mathbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

- a nonterminal transition represents *exactly* its derived context-free language

Nonterminal Transitions

- $\mathcal{SF}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\equiv)$
- look for two different bracketed sentential forms in $\mathcal{L}(\Gamma/\equiv)$

$$\mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \mathrm{d}_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle \mathrm{r}_2\ \mathrm{r}_{10}$$
$$\mathrm{d}_2\ \mathrm{d}_{10}\ \mathit{vid}\ \textbf{as}\ \langle \mathit{pat} \rangle\ \mathrm{r}_{10}\ :\ \langle \mathit{ty} \rangle\ \mathrm{r}_2$$

- a nonterminal transition represents *exactly* its derived context-free language

Nonterminal Transitions

- $\mathcal{SF}(\mathcal{G}_b) \subseteq \mathcal{L}(\Gamma/\!\equiv)$
- look for two different bracketed sentential forms in $\mathcal{L}(\Gamma/\!\equiv)$

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

- a nonterminal transition represents *exactly* its derived context-free language

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty}\rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty}\rangle\ r_2$$

epsilon: mae

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty}\rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty}\rangle\ r_2$$

epsilon: mae

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty} \rangle\ r_2$$

epsilon: mae

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty}\rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \mathbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty}\rangle\ r_2$$

shift: mas

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty}\rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty}\rangle\ r_2$$

shift: mas

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \textit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \textit{vid}\ r_3\ :\langle \textit{ty}\rangle\ r_2\ r_{10}$$

$$d_2\ d_{10}\ \textit{vid}\ \textbf{as}\ d_1\ d_3\ \textit{vid}\ r_3\ r_1\ r_{10}\ :\langle \textit{ty}\rangle\ r_2$$

epsilon: mae

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ d_1\ d_3\ \mathit{vid}\ r_3\ :\langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ d_1\ d_3\ \mathit{vid}\ r_3\ r_1\ r_{10}\ :\langle \mathit{ty} \rangle\ r_2$$

nothing!

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

d_{10} *vid* **as** d_2 d_1 ⟨*atpat*⟩ r_1 : ⟨*ty*⟩ r_2 r_{10}

d_2 d_{10} *vid* **as** d_1 ⟨*atpat*⟩ r_1 r_{10} : ⟨*ty*⟩ r_2

shift: mas

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ d_1\ \langle \mathit{atpat} \rangle\ r_1\ :\langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \mathbf{as}\ d_1\ \langle \mathit{atpat} \rangle\ r_1\ r_{10}\ :\langle \mathit{ty} \rangle\ r_2$$

nothing!

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ \langle \mathit{pat} \rangle\ : \langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ : \langle \mathit{ty} \rangle\ r_2$$

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \mathbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

conflict: mac

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

d_{10} *vid* **as** d_2 $\langle pat \rangle$: $\langle ty \rangle$ r_2 r_{10}
d_2 d_{10} *vid* **as** $\langle pat \rangle$ r_{10} : $\langle ty \rangle$ r_2

shift: mas

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \textbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \textbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

reduce: mar

Mutual Accessibility Relations

- between pairs of states of $\Gamma/\!\equiv$, (q_1, q_2)
- synchronized left-to-right walks from an initial pair (q_s, q_s)

$$d_{10}\ \mathit{vid}\ \mathbf{as}\ d_2\ \langle \mathit{pat} \rangle\ :\ \langle \mathit{ty} \rangle\ r_2\ r_{10}$$
$$d_2\ d_{10}\ \mathit{vid}\ \mathbf{as}\ \langle \mathit{pat} \rangle\ r_{10}\ :\ \langle \mathit{ty} \rangle\ r_2$$

conflict: mac

NU(≡)

- ma=mas ∪ mae ∪ mac ∪ mar
- $\mathcal{G}$ is *noncanonically unambiguous* if there does not exist a relation (q_s, q_s) ma* (q_f, q_f) that uses mac at some step
- Computation in $\mathcal{O}(|\Gamma/{\equiv}|^2)$

Comparisons

- Regular Unambiguity $RU(\equiv)$
- Bounded-length detection schemes
- $LR(k)$ and LR-Regular ($LR(\Pi)$)
- Horizontal and vertical ambiguity ($HVRU(\equiv)$)

Bounded-length detection

[Gorn, 1963, Cheung and Uzgalis, 1995, Schröer, 2001, Jampana, 2005]

- generate sentences
- not conservative
- prefix_m prevents from false positives in sentences of length $< m$
- need to generate a^{2^n+1} to find $\mathcal{G}_4^n$ ambiguous, but $\mathcal{G}_4^n \notin \mathrm{NU}(\text{item}_0)$

$$S \to A \mid B_n a,\ A \to Aaa \mid a,$$
$$B_1 \to aa,\ B_2 \to B_1 B_1, \ldots,\ B_n \to B_{n-1} B_{n-1} \qquad (\mathcal{G}_4^n)$$

LR(k) and LR-Regular

[Knuth, 1965, Hunt III et al., 1975, Čulik and Cohen, 1973, Heilbrunner, 1983]

- conservative tests
- define item_Π s.t. $\mathrm{LR}(\Pi) \subset \mathrm{NU}(\mathsf{item}_\Pi)$
- need a LR(2^n) test to prove $\mathcal{G}_3^n$ unambiguous, but $\mathcal{G}_3^n \in \mathrm{NU}(\mathsf{item}_0)$

$$S \to A \mid B_n,\ A \to Aaa \mid a,$$
$$B_1 \to aa,\ B_2 \to B_1B_1, \ldots,\ B_n \to B_{n-1}B_{n-1} \qquad (\mathcal{G}_3^n)$$

Implementation

- For the whole SML grammar:
  - conflicts in the LALR(1) parser

    `sml.y: conflicts: 223 shift/reduce, 35 reduce/reduce`
  - Our tool:

    `89 potential ambiguities with LR(1) precision detected`

- For the SML grammar fragment:
- Benchmark: $\mathrm{NU}(\mathsf{item}_1)$ correctly identifies 87% of our unambiguous grammars—73% of the non-LALR(1) ones

Summary

- conservative ambiguity detection
- provably better than several other techniques
- also experimentally better

Disjointness in PEGs

- $A \leftarrow \alpha_1/\alpha_2, B \leftarrow \alpha_2/\alpha_1$
- disjointness: $\mathcal{L}(A) = \mathcal{L}(B)$
- semi-disjointness: $\mathcal{L}(\alpha_1) \cap \mathcal{L}(\text{SPrefix}(\alpha_2)) = \emptyset$
- general semi-disjointness: $\mathcal{L}(\mathcal{G}[A \leftarrow\!\!\dashv \alpha_1]) \cap \mathcal{L}(\mathcal{G}[A \leftarrow\!\!\dashv \text{SPrefix}(\alpha_2)]) = \emptyset$

Context-Free and Regular Approximations

- context-free equivalent of $A \leftarrow \alpha_1 / \ldots / \alpha_n$:
  $A \rightarrow \alpha_1 | \ldots | \alpha_n$
- $\mathcal{L}_{\text{PEG}}(\mathcal{G}) \subseteq \mathcal{L}_{\text{CFG}}(\mathcal{G})$
- regular approximations, again

Verifying PEGs

$$\langle pat \rangle \leftarrow \langle atpat \rangle\ \langle tyop \rangle\ /\ vid\ \langle tyop \rangle\ \textbf{as}\ \langle pat \rangle\ \langle tyop \rangle$$
$$\langle atpat \rangle \leftarrow vid$$

accessibility, again:

$$\mathrm{d}_1\ \mathrm{d}_3\ vid\ \mathrm{r}_3\ \langle tyop \rangle\ \mathrm{r}_1$$
$$\mathrm{d}_2\ vid\ \langle tyop \rangle\ \textbf{as}\ \langle pat \rangle\ \langle tyop \rangle\ \mathrm{r}_2$$

Verifying PEGs

$$\langle pat\rangle \leftarrow \langle atpat\rangle\ \langle tyop\rangle\ /\ vid\ \langle tyop\rangle\ \textbf{as}\ \langle pat\rangle\ \langle tyop\rangle$$
$$\langle atpat\rangle \leftarrow vid$$

accessibility, again:

$$\mathrm{d}_1\ \mathrm{d}_3\ \color{red}{vid}\ \mathrm{r}_3\ \color{red}{\langle tyop\rangle\ \mathrm{r}_1}$$
$$\mathrm{d}_2\ \color{red}{vid\ \langle tyop\rangle}\ \textbf{as}\ \langle pat\rangle\ \langle tyop\rangle\ \mathrm{r}_2$$

Summary

- need for grammar verification
- conservative tests

M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Comput. Sci.*, 292(1):45–63, 2003. ISSN 0304-3975. doi: 10.1016/S0304-3975(01)00214-6. Selected papers in honor of Jean Berstel.

A. Birman and J. D. Ullman. Parsing algorithms with backtrack. *Information and Control*, 23(1):1–34, 1973. ISSN 0019-9958. doi: 10.1016/S0019-9958(73)90851-6.

C. Brabrand, R. Giegerich, and A. Møller. Analyzing ambiguity of context-free grammars. In J. Holub and J. Žďárek, editors, *CIAA'07*, volume 4783 of *Lecture Notes in Computer Science*, pages 214–225. Springer, 2007. ISBN 978-3-540-76335-2. doi: 10.1007/978-3-540-76336-9_21.

D. G. Cantor. On the ambiguity problem of Backus systems. *J. ACM*, 9(4): 477–479, 1962. ISSN 0004-5411. doi: 10.1145/321138.321145.

B. S. N. Cheung and R. C. Uzgalis. Ambiguity in context-free grammars. In *SAC'95*, pages 272–276. ACM Press, 1995. ISBN 0-89791-658-1. doi: 10.1145/315891.315991.

N. Chomsky and M. P. Schützenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirshberg, editors, *Computer Programming and Formal Systems*, Studies in Logic, pages 118–161. North-Holland Publishing, 1963.

S. Crespi Reghizzi and G. Psaila. Grammar partitioning and modular deterministic parsing. *Computer Languages*, 24(4):197–227, 1998. ISSN 0096-0551. doi: 10.1016/S0096-0551(98)00012-5.

K. Čulik and R. Cohen. LR-Regular grammars—an extension of LR(k) grammars. *J. Comput. Syst. Sci.*, 7(1):66–96, 1973. ISSN 0022-0000. doi: 10.1016/S0022-0000(73)80050-9.

S. Even. On information lossless automata of finite order. *IEEE Transactions on Electronic Computers*, EC-14(4):561–569, 1965. ISSN 0367-7508. doi: 10.1109/PGEC.1965.263996.

B. Ford. Parsing expression grammars: a recognition-based syntactic foundation. In *POPL'04*, pages 111–122. ACM Press, 2004. ISBN 1-58113-729-X. doi: 10.1145/964001.964011.

S. Gorn. Detection of generative ambiguities in context-free mechanical languages. *J. ACM*, 10(2):196–208, 1963. ISSN 0004-5411. doi: 10.1145/321160.321168.

R. Grimm. Better extensibility through modular syntax. In *PLDI'06*, pages 38–51. ACM Press, 2006. ISBN 1-59593-320-4. doi: 10.1145/1133981.1133987.

S. Heilbrunner. Tests for the LR-, LL-, and LC-Regular conditions. *J. Comput. Syst. Sci.*, 27(1):1–13, 1983. ISSN 0022-0000. doi: 10.1016/0022-0000(83)90026-0.

H. B. Hunt III, T. G. Szymanski, and J. D. Ullman. On the complexity of LR(k) testing. *Commun. ACM*, 18(12):707–716, 1975. ISSN 0001-0782. doi: 10.1145/361227.361232.

S. Jampana. Exploring the problem of ambiguity in context-free grammars. Master's thesis, Oklahoma State University, July 2005. URL `http://e-archive.library.okstate.edu/dissertations/AAI1427836/`.

P. Klint, R. Lämmel, and C. Verhoef. Toward an engineering discipline for grammarware. *ACM Transactions on Software Engineering and Methodology*, 14(3):331–380, 2005. ISSN 1049-331X. doi: 10.1145/1072997.1073000.

D. E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, 1965. ISSN 0019-9958. doi: 10.1016/S0019-9958(65)90426-2.

R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The definition of Standard ML*. MIT Press, revised edition, 1997. ISBN 0-262-63181-4.

F. W. Schröer. AMBER, an ambiguity checker for context-free grammars. Technical report, compilertools.net, 2001. URL `http://accent.compilertools.net/Amber.html`.

E. Visser. *Syntax Definition for Language Prototyping*. PhD thesis, Sept. 1997. URL `http://citeseer.ist.psu.edu/visser97syntax.html`.