Generalised parsing with Binary Subtree Representations (BSR)

L. Thomas van Binsbergen,
Elizabeth Scott, and Adrian Johnstone

Royal Holloway, University of London
ltvanbinsbergen@acm.org

24 April, 2019

Research output: SLE 2018, SCP 2019 vol. 175
Hackage packages: `gll`, `happy` (in development)
Thesis: `http://ltvanbinsbergen.nl/thesis`

Past research

- Alternative descriptions of GLL parsing:
  - Purely functional GLL parsing
  - GLL parsing in parser combinators

All developed parser algorithms output binary subtree representation (BSR) sets

Future work

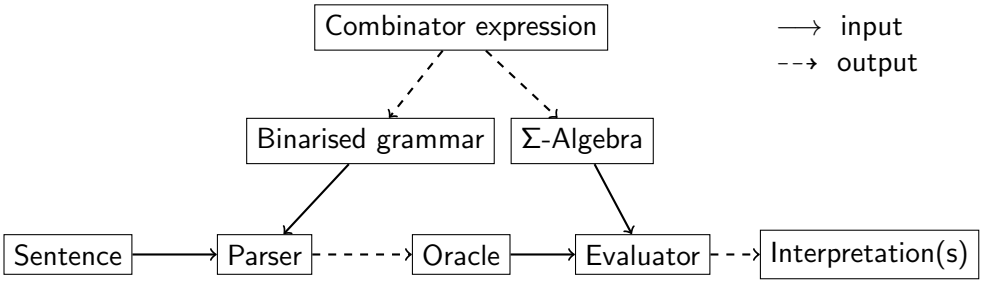
- Generating GLL parsers from:
  - Parameterised grammar descriptions (Happy)
  - Generating GLL parsers from CBS syntax descriptions
- Expressing disambiguation strategies on BSR sets

Simple, efficient, sound and complete combinator parsing for all context-free grammars, using an oracle

Tom Ridge

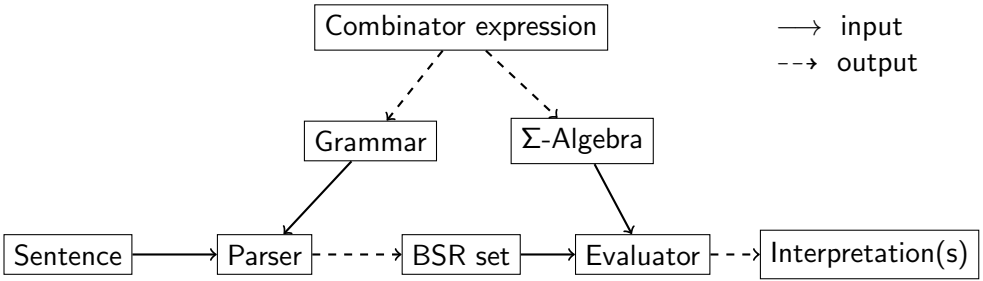
University of Leicester

Library architecture (Ridge 2014)



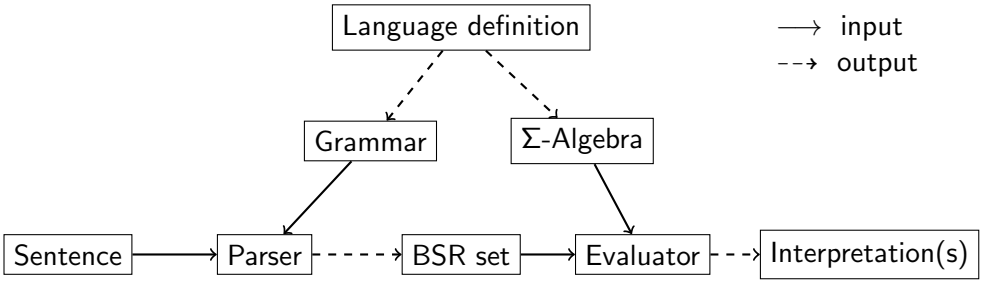
- Parser is replaceable
- Similar suggestion by [Ljunglöf, 2002]

Library architecture (SLE 2018)



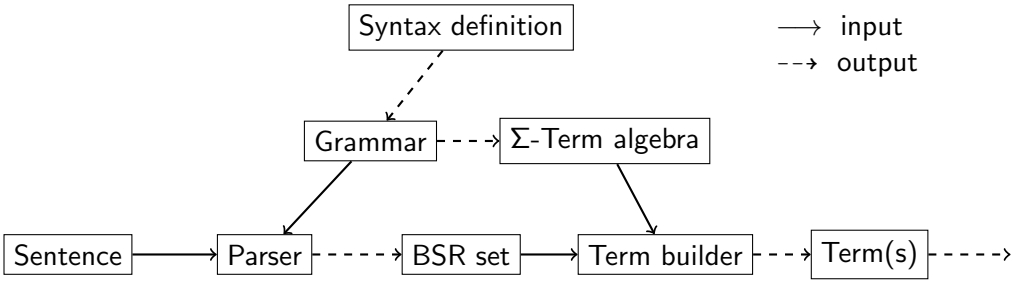
- Parser is replaceable
- Binary Subtree Representation (BSR) sets for arbitrary CFGs

General architecture



- Parser is replaceable
- Binary Subtree Representation (BSR) sets for arbitrary CFGs

Term construction



- Parser is replaceable
- The term builder might construct terms that share common sub-terms

Proposal

Let parsers produce traces that contain sufficient information to reproduce derivations and delay term construction or interpretation to a separate phase

Benefits

- Separation of concerns
- Simplifies definition of the parser
- Parser improvements/optimisations do not influence subsequent phases

Parsing

Grammar:	$\Gamma = \langle N, T, P, Z \rangle$
Input sentence:	$t_0 \dots t_{m-1}$
Grammar slots:	$G(\Gamma) = \{X \to \alpha \bullet \beta \mid X \to \alpha\beta \in P\}$
Items:	$\mathcal{I}(\Gamma, t_0 \dots t_{m-1}) = \{\langle g, l, k \rangle \mid g \in G(\Gamma), 0 \leqslant l \leqslant k \leqslant m\}$
Valid items:	$\mathcal{U}(\Gamma, t_0 \dots t_{m-1}) = \dots$

Parsing schemata (Sikkel 1998)

Valid items are inferable according to some inference relation

The inference relation captures a parsing algorithm conceptually (not operationally)

Generalised top-down parsing (canonical Earley)

$$\frac{Z \to \delta \in P}{\langle Z \to \bullet\delta, 0, 0\rangle \in \mathcal{U}} \quad \text{(init)}$$

$$\frac{\langle X \to \alpha\bullet t\beta, l, k\rangle \in \mathcal{U} \quad t = t_k}{\langle X \to \alpha t\bullet\beta, l, k+1\rangle \in \mathcal{U}} \quad \text{(scanner)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \quad \text{(predictor)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad \langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U}}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \quad \text{(completer)}$$

BSR set construction

$$\langle X \to \alpha \bullet s \beta, l, k \rangle \in \mathcal{U} \quad \& \quad \langle X \to \alpha s \bullet \beta, l, r \rangle \in \mathcal{U}$$

gives

$$\langle X \to \alpha s \bullet \beta, l, k, r \rangle \in \Upsilon$$

$$\langle Y \to \bullet, l, l \rangle \in \mathcal{U}$$

gives

$$\langle Y \to \bullet, l, l, l \rangle \in \Upsilon$$

```
program chart parser
begin
    create initial chart and agenda;
    while agenda is not empty
    do
        delete some (arbitrarily chosen) current item from agenda
        add current to chart;
        for each item that can be recognized by current in
        combination with other items in chart
        do
            if item is neither in chart nor in agenda
            then add item to agenda fi
        od
    od
end.
```

The chart parser algorithm.

(Sikkel 1998).

Implementing canonical Earley

Earley's algorithm: process items in ascending order of k (by executing *scanner* last)

The *completer* on grammars without nullable nonterminals

$$\frac{\langle X \to \alpha \bullet Y \beta, l, k\rangle \in \mathcal{U} \quad \langle Y \to \delta \bullet, k, r\rangle \in \mathcal{U}}{\langle X \to \alpha Y \bullet \beta, l, r\rangle \in \mathcal{U}} \qquad \text{(completer)}$$

When processing $\langle X \to \alpha \bullet Y \beta, l, k\rangle$, no items $\langle Y \to \delta \bullet, k, r\rangle$ will be discovered yet

When processing $\langle Y \to \delta \bullet, k, r\rangle$, all $\langle X \to \alpha \bullet Y \beta, l, k\rangle$ are discovered when $k = r$

GLL parsing: process items in any order with special data structures that are used instead of 'searching through' $\mathcal{U}$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y,k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad r \in \mathtt{pop}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p \quad \emptyset = \mathtt{pop}(Y,k)}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U}}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \quad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U}}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \quad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \quad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y, k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \quad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U}}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \quad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in \boldsymbol{p}}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \quad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y, k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y, k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U}}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \qquad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y,k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad r \in \mathtt{pop}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U}} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \qquad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y,k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad r \in \mathtt{pop}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \qquad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y,k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad r \in \mathtt{pop}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p \quad \emptyset = \mathtt{pop}(Y,k)}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U}} \qquad \text{(predictor/descend)}$$

GLL parsing

GLL introduces a GSS and a pop-set for both directions of the *completer* action

$$\frac{\langle Y \to \delta\bullet, k, r\rangle \in \mathcal{U} \quad \langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [r \in \mathtt{pop}(Y,k)]} \qquad \text{(completer (1)/ascend)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad r \in \mathtt{pop}(Y,k)}{\langle X \to \alpha Y\bullet\beta, l, r\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(completer (2)/skip)}$$

$$\frac{\langle X \to \alpha\bullet Y\beta, l, k\rangle \in \mathcal{U} \quad Y \to \delta \in p \quad \emptyset = \mathtt{pop}(Y,k)}{\langle Y \to \bullet\delta, k, k\rangle \in \mathcal{U} \quad [\langle X \to \alpha Y\bullet\beta, l\rangle \in \mathtt{gss}(Y,k)]} \qquad \text{(predictor/descend)}$$

GLL parsing summary

- Items can be processed through a worklist in any order
- For each item being processed, there is at most one applicable rule
  Possible actions: *descend*, *ascend*, *skip*, *ascend*
- If a rule is applicable for multiple instantiations, apply it for all instantiations 'simultaneously'

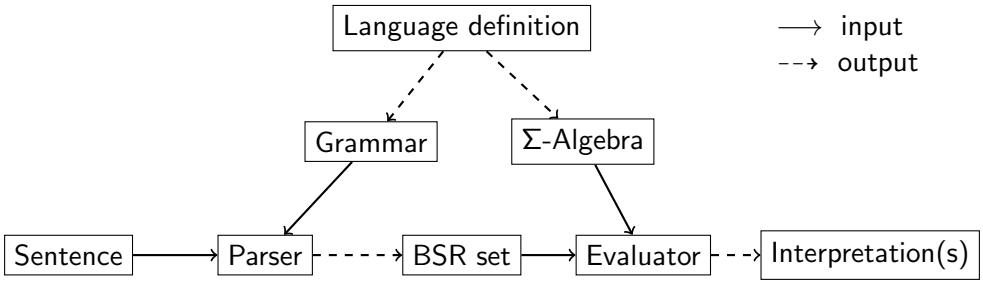
Correctness

Termination	There is a finite number of items: $O(m^2)$
Soundness	Derived from canonical Earley
Completeness	Derived from: $skip + ascend = completer$

Summary - Part 1

- Any parsing algorithm that specialises canonical Earley can yield BSR sets
- GLL parsing specialises Earley with data structures for *completer*

General architecture (recap)



Language definitions

Definition

A *language definition* consists of a CFG $\Gamma = \langle N, T, P, Z \rangle$, a family of sets $\{\tau_s\}_{s \in N \cup T}$ with $\tau_t = \{t\}$ for $t \in T$, and a family of higher-order functions $\{\mathcal{A}_p\}_{p \in P}$ with

$$\mathcal{A}_{s_{n+1} \to s_1 \ldots s_n} : \tau_{s_1} \to \ldots \to \tau_{s_n} \to \tau_{s_{n+1}} \qquad \text{(type also denoted } \tau_{s_1 \ldots s_{n+1}}\text{)}$$

Language definitions

Definition

A *language definition* consists of a CFG $\Gamma = \langle N, T, P, Z \rangle$, a family of sets $\{\tau_s\}_{s \in N \cup T}$ with $\tau_t = \{t\}$ for $t \in T$, and a family of higher-order functions $\{\mathcal{A}_p\}_{p \in P}$ with

$$\mathcal{A}_{s_{n+1} \to s_1 \ldots s_n} : \tau_{s_1} \to \ldots \to \tau_{s_n} \to \tau_{s_{n+1}} \qquad \text{(type also denoted } \tau_{s_1 \ldots s_{n+1}}\text{)}$$

Partial application

The family $\mathcal{A}$ is extended with a function for each slot $g \in G(\Gamma)$

$$\mathcal{A}_{X \to \bullet\beta} : \tau_\beta X$$
$$\mathcal{A}_{X \to \bullet\beta} = \mathcal{A}_{X \to \beta}$$

$$\mathcal{A}_{X \to \alpha s \bullet \beta} : (\tau_{s\beta} X \times \tau_s) \to \tau_\beta X$$
$$\mathcal{A}_{X \to \alpha s \bullet \beta}(f, v) = f(v)$$

Canonical BSR-based Evaluation

For each $s \in N \cup T$, a function $\mathcal{E}_s : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_s)$ is defined and for each $g \in G(\Gamma)$ with $g = X \to \alpha \bullet \beta$ a function $\mathcal{E}_g : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_{\beta X})$

Canonical BSR-based Evaluation

For each $s \in N \cup T$, a function $\mathcal{E}_s : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_s)$ is defined and for each $g \in G(\Gamma)$ with $g = X \to \alpha \bullet \beta$ a function $\mathcal{E}_g : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_{\beta X})$

$$\mathcal{E}_t(\Upsilon, l, r) = \{t \mid r = l + 1\}$$

Canonical BSR-based Evaluation

For each $s \in N \cup T$, a function $\mathcal{E}_s : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_s)$ is defined and for each $g \in G(\Gamma)$ with $g = X \to \alpha \bullet \beta$ a function $\mathcal{E}_g : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_{\beta X})$

$$\mathcal{E}_t(\Upsilon, l, r) = \{t \mid r = l + 1\}$$

$$\mathcal{E}_X(\Upsilon, l, r) = \{v \mid X \to \beta \in P, v \in \mathcal{E}_{X \to \beta \bullet}(\Upsilon, l, r)\}$$

Canonical BSR-based Evaluation

For each $s \in N \cup T$, a function $\mathcal{E}_s : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_s)$ is defined and for each $g \in G(\Gamma)$ with $g = X \to \alpha \bullet \beta$ a function $\mathcal{E}_g : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_{\beta X})$

$$\mathcal{E}_t(\Upsilon, l, r) = \{t \mid r = l + 1\}$$

$$\mathcal{E}_X(\Upsilon, l, r) = \{v \mid X \to \beta \in P, v \in \mathcal{E}_{X \to \beta \bullet}(\Upsilon, l, r)\}$$

$$\mathcal{E}_{X \to \bullet \beta}(\Upsilon, l, r) = \{\mathcal{A}_{X \to \bullet \beta} \mid l = r\}$$

Canonical BSR-based Evaluation

For each $s \in N \cup T$, a function $\mathcal{E}_s : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_s)$ is defined and for each $g \in G(\Gamma)$ with $g = X \to \alpha \bullet \beta$ a function $\mathcal{E}_g : \hat{\Upsilon} \times \mathbb{N} \times \mathbb{N} \to \mathcal{P}(\tau_{\beta X})$

$$\mathcal{E}_t(\Upsilon, l, r) = \{t \mid r = l + 1\}$$

$$\mathcal{E}_X(\Upsilon, l, r) = \{v \mid X \to \beta \in P, v \in \mathcal{E}_{X \to \beta \bullet}(\Upsilon, l, r)\}$$

$$\mathcal{E}_{X \to \bullet \beta}(\Upsilon, l, r) = \{\mathcal{A}_{X \to \bullet \beta} \mid l = r\}$$

$$\begin{aligned}\mathcal{E}_{X \to \alpha s \bullet \beta}(\Upsilon, l, r) = \{\mathcal{A}_{X \to \alpha s \bullet \beta}(f, v) \mid \langle X \to \alpha s \bullet \beta, l, k, r\rangle \in \Upsilon \\ , v \in \mathcal{E}_s(\Upsilon, k, r) \\ , f \in \mathcal{E}_{X \to \alpha \bullet s \beta}(\Upsilon, l, k)\}\end{aligned}$$

Term construction (without sharing)

Terms

Induced by a CFG is a family of sets of terms $\{\mathcal{T}_s\}_{s \in N \cup T}$ where $\mathcal{T}_t = \{t\}$ (for $t \in T$) and $\mathcal{T}_X$ (for $X \in N$) is defined as the smallest set such that:

$$\frac{X \to s_1 ... s_n \in P \quad t_1 \in \mathcal{T}_{s_1} \ldots t_n \in \mathcal{T}_{s_n}}{\langle X, t_1 ... t_n \rangle \in \mathcal{T}_X}$$

Term construction

Given a CFG, form a language definition by taking $\tau_s = \mathcal{T}_s$ (for all $s \in N \cup T$) and by defining $\mathcal{A}_p$ as follows (for all $p \in P$ with $p = X \to s_1 ... s_n$)

$$\mathcal{A}_{X \to s_1 ... s_n}(t_1) \ldots (t_n) = \langle X, t_1 ... t_n \rangle$$

Evaluator modifications for ambiguity reduction

1. General filtering of nonterminal results
2. Removing 'invalid' interpretations (invalid according to some static semantics)

   ...
3. Removing 'non-minimal' terms (or their interpretations) for cyclic grammars
4. Associativity of operators
5. Precedence of operators
6. Follow-restrictions
7. Longest/shortest-match

2. Removing invalid interpretations

Let $\tau^o_s = \{\textbf{none}\} \cup \{\textbf{some}(v) \mid v \in \tau_s\}$ and let $\tau^o_{s_1 \ldots s_n}$ denote $\tau_{s_1} \to \ldots \to \tau_{s_{n-1}} \to \tau^o_{s_n}$

Language definitions

The actions $\mathcal{A}_p$ of a language definition may now fail, as is reflected in the types:

$$\mathcal{A}_{s_{n+1} \to s_1 \ldots s_n} : \tau^o_{s_1 \ldots s_{n+1}}$$
$$\mathcal{A}_{s_{n+1} \to \bullet s_1 \ldots s_n} : \tau^o_{s_1 \ldots s_{n+1}}$$
$$\mathcal{A}_{s_{n+1} \to \alpha s_0 \bullet s_1 \ldots s_n} : (\tau^o_{s_0 \ldots s_{n+1}} \times \tau_{s_0}) \to \tau^o_{s_1 \ldots s_{n+1}}$$

Update the definition of $\mathcal{E}_X$:

$$\mathcal{E}_X(\Upsilon, I, r) = \{v \mid \langle X \to \beta\bullet, I, k, r\rangle \in \Upsilon, \textbf{some}(v) \in \mathcal{E}_{X \to \beta\bullet}(\Upsilon, I, r)\}$$

Arithmetic expression example

Language definitions

Let language definitions provide a family of functions $\{\mathcal{F}_X\}_{X \in N}$ with $\mathcal{F}_X : \mathcal{P}(\tau_X) \to \mathcal{P}(\tau_X)$ s.t. for all $\phi_X \subseteq \mathcal{P}(\tau_X)$ it holds that $\mathcal{F}_X(\phi_X) \subseteq \phi_X$.

Update the definition of $\mathcal{E}_X$:

$$\mathcal{E}_X(\Upsilon, l, r) = \mathcal{F}_X(\{v \mid \langle X \to \beta\bullet, l, k, r\rangle \in \Upsilon, \mathbf{some}(v) \in \mathcal{E}_{X \to \beta\bullet}(\Upsilon, l, r)\})$$

Overloading example

Cyclic grammars

A grammar is cyclic if it has a nonterminal that can derive itself via one or more steps

Definition

A term is non-minimal if a subterm t can be replaced by a term t' for which holds that t and t' have the same label and yield, and t' is a subterm of t.

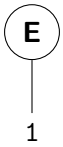
- A sentence that admits a non-minimal term has infinitely many interpretations
- Cyclic grammars are the only grammars that admit non-minimal terms

Cyclic grammars - Example

$$\mathbf{E} \quad ::= \quad \mathbf{E}\,\mathbf{E}\,\mathbf{E} \mid \texttt{"1"} \mid \epsilon$$

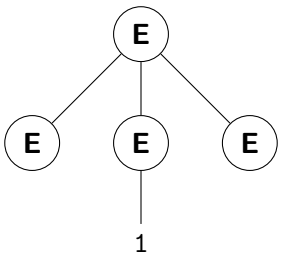
Cyclic grammars - Example

$$\mathbf{E} \quad ::= \quad \mathbf{E}\,\mathbf{E}\,\mathbf{E} \mid \texttt{"1"} \mid \epsilon$$



Cyclic grammars - Example

$$\mathbf{E} \quad ::= \quad \mathbf{E}\,\mathbf{E}\,\mathbf{E} \mid \texttt{"1"} \mid \epsilon$$



Remember the nonterminal labels of ancestor nodes with the same extents in $\mathcal{C}$

$$\mathcal{E}_t(\Upsilon, l, r, \mathcal{C}) = \{t \mid r = l + 1\}$$

$$\mathcal{E}_X(\Upsilon, l, r, \mathcal{C}) = \{v \mid X \notin \mathcal{C}, X \to \beta \in P, v \in \mathcal{E}_{X \to \beta\bullet}(\Upsilon, l, r, \mathcal{C} \cup \{X\})\}$$

$$\mathcal{E}_{X \to \bullet\beta}(\Upsilon, l, r, \mathcal{C}) = \{\mathcal{A}_{X \to \bullet\beta} \mid l = r\}$$

$$\begin{aligned}\mathcal{E}_{X \to \alpha s\bullet\beta}(\Upsilon, l, r, \mathcal{C}) = \{\mathcal{A}_{X \to \alpha s\bullet\beta}(f, v) \mid \langle X \to \alpha s\bullet\beta, l, k, r\rangle \in \Upsilon \\ , v \in \mathcal{E}_s(\Upsilon, k, r, \{Y \mid k = l, Y \in \mathcal{C}\}) \\ , f \in \mathcal{E}_{X \to \alpha\bullet s\beta}(\Upsilon, l, k, \{Y \mid k = r, Y \in \mathcal{C}\})\}\end{aligned}$$

Evaluator modifications for ambiguity reduction

1. General filtering of nonterminal results
2. Removing 'invalid' interpretations (invalid according to some static semantics)

   ...
3. Removing 'non-minimal' terms (or their interpretations) for cyclic grammars
4. Associativity of operators
5. Precedence of operators
6. Follow-restrictions
7. Longest/shortest-match

Summary - Part 2

- The binary subtree representation can be the basis of a separate evaluation phase
- General and specific disambiguation strategies can manifest themselves in the evaluation phase

Research Questions

- Is it feasible to disambiguate with the more general strategies only?
- How to avoid hard-coding strategies into the evaluation algorithm?
- With what sort of technique could we describe strategies at BSR element level, without interfering (much) with the evaluator?

Scales example

Generalised parsing with Binary Subtree Representations (BSR)

L. Thomas van Binsbergen,
Elizabeth Scott, and Adrian Johnstone

Royal Holloway, University of London
ltvanbinsbergen@acm.org

24 April, 2019

Research output: SLE 2018, SCP 2019 vol. 175
Hackage packages: `gll`, `happy` (in development)
Thesis: `http://ltvanbinsbergen.nl/thesis`

Parser Combinators	`parsec`
	`UU-lib`
	...

Explicit Nonterminals	Scheme recognisers (Johnson 1995) `Meerkat` (Izmaylova/Afroozeh 2015/16)
Grammar Combinators	`P3` (Ridge 2014) GLL.Combinators (2015/16) `grammar-combinators` (Devriese 2011/12)
Meta-Programming	`BNFC-meta` (Duregard 2011)

Parser Generators	`Bison`
	`yacc`
	`Happy`
	...