Modelling Parsing and Unparsing

Vadim Zaytsev
University of Amsterdam

Anya Helene Bagge
University of Bergen

Parsing@SLE
2014-09-14
String

\[ f \ arg = \ arg + 1; \]

Parse Tree

Token List

\[ f \ arg \ - \ arg \ + \ 1 \ ; \]
The Megamodel

Lex (lexical model) → Ast (abstract syntax tree) → Dia (diagram)
TTk (typed tokens) → Cst (concrete syntax tree) → Gra (graph model)
Tok (tokens) → Ptr (parse tree) → Dra (vector drawing)
Str (string) → For (parse forest) → Pic (rasterised picture)

Tokenise → concat → parse → unparse

Parseforest → Parsetree → Concrete syntaxtree

Structural editing

M2m transformation

Flatten → extract → strip → format → strip → format

Text editing

Data editing

Raw → Layoutless → Abstract

Layout → Raw → Textual

Structured

Graphical
Classic (Lex+Yacc) Parsing

Lex (lexical model)

Ast (abstract syntax tree)

Dia (diagram)

TTk (typed tokens)

Cst (concrete syntax tree)

Gra (graph model)

Tok (tokens)

Ptr (parse tree)

Dra (vector drawing)

Str (string)

For (parse forest)

Pic (rasterised picture)

scan

parse

Zaytsev & Bagge (UvA/UiB)
Code Formatting

Abstract

Lex
(lexical model)

Ast
(abstract syntax tree)

Dia
(diagram)

Layoutless

TTk
(typed tokens)

Cst
(concrete syntax tree)

Gra
(graph model)

Layout

Tok
(tokens)

Ptr
(parse tree)

Dra
(vector drawing)

Raw

Str
(string)

For
(parse forest)

Pic
(rasterised picture)

Textual

Structured

Graphical

concat
unparse
format
concat

Zaytsev & Bagge (UvA/UiB)
Rascal
Lowered Transformations

\[ \text{Ast} \xrightarrow{\text{transform}} \text{Ast} \]
\[ \text{Cst} \xleftarrow{\text{implode}} \text{Cst} \]
\[ \text{Cst} \xrightarrow{\text{transform}} \text{Cst} \]

Zaytsev & Bagge (UvA/UiB)
Lowered Transformations

```
Ast -- transform --> Ast
      \      /    /
      implode explode

Cst -- transform --> Cst
      /      \    /
strip    format

Ptr -- transform --> Ptr
```
Lowered Transformations

Ast → transform → Ast
Ast ↓ implode ↓ Ast

Cst ← transform ← Cst
Cst ↑ transform ↑ Cst

Ptr ← transform ← Ptr
Ptr ↓ unparse ↓ Ptr

Str ← transform ← Str
Str ↑ parse ↑ Str
Composing Transformations

Abstract
- Lex (lexical model)
- Ast (abstract syntax tree)
- Dia (diagram)

Layoutless
- TTk (typed tokens)
- Cst (concrete syntax tree)

Layout
- Tok (tokens)
- Ptr (parse tree)

Raw
- Str (string)
- For (parse forest)

Textual
- concat
- parse

Structured
- format

Graphical
- unpars
- Dra (vector drawing)

Zaytsev & Bagge (UvA/UiB)
Modelling Parsing and Unparsing
2014-09-14 8 / 10
Composing Transformations

Lex (lexical model) → Ast (abstract syntax tree) → Dia (diagram)

TTk (typed tokens) → unpars → Cst (concrete syntax tree) → strip

Tok (tokens) → format → TTk (typed tokens)

Str (string) → parse → For (parse forest) → filter

Pic (rasterised picture) → Gra (graph model) → Dra (vector drawing)

Textual → Structured → Graphical
Is it Useful?

- Explain representations and transformations to students
- Explain what you mean with “parsing” in your paper
- Simplify dealing with bidirectional transformations
- Amaze your friends with fancy graphic models in your papers
Conclusion

- We have identified SLE artifacts and translations between them

Prototype w/transformations:
  https://github.com/grammarware/bx-parsing

The full paper: