UPTR - a simple parse tree representation format

Jurgen Vinju

Software Transformation Systems

October 22, 2006

Quality of Software Transformation Systems

- Quality is a subjective concept
    - Satisfaction of user requirements
    - Which STS is best for my task?
    - Which STS does better for these kinds of tasks?
- STS's are hard to compare
    - Terminology and concepts differ
    - Pretty different goals and requirements
    - Comparing software is a complicated task anyway
- Simple questions:
    - What things are we comparing?
    - On what attributes are they compared?
    - How do we compare precisely and honestly?

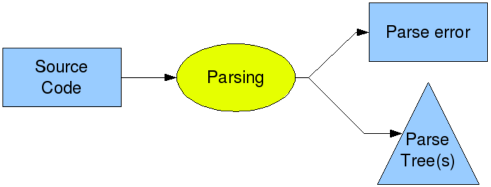
Some Questions on Parsing

- What does this STS consider to be parsing?
- Is lexical analysis included/excluded?
- What is the input and output of parsing?
- Is the output an AST? Parse tree? Something else?
- Are the source code comments still in there? Whitespace?
- Is the output of this parser "correct"?
- Is this the ISO standard interpretation for this C++ program?
- Is this the GNU interpretation of this C program?
- How did this parser resolve the dangling else issue?
- How fast is this parser?

Two pragmatical steps

1. Consensus on the input and output of parsing
    - Common, more precise, terminology
    - Measurable parsing process
2. Standard output (file) format for parsing
    - Enables validation of parsers by comparing parse trees
    - Enables reuse of parsers (UPTR as input format)
    - Enables reuse of backends (UPTR as output format)

The input, process, and output



- Input a program file
- . . . Including lexical (regular) analysis
- . . . Including syntax (context-free) analysis
- . . . Including all kinds of (context-sensitive) disambiguation
- . . . Excluding abstraction
- . . . Excluding simplification/normalization
- Output parse trees

UPTR

- Universal Parse Tree Representation
- An exact representation of one or more derivations
  - for a certain input program
  - for a certain parser
- What is in a UPTR file?
  - Regular expressions and context-free grammar rules (nodes)
  - The characters of the input file (leaves)
  - Ambiguity clusters (nodes)
  - Cyclic references (leaves)

UPTR properties

- Old
    - More than 9 years of experience
    - ASF+SDF, StrategoXT, ELAN, Action Notation, ...
- Verbose
    - Contains the full grammar and regular expressions
    - Can contain full lexical structure
    - Can contain all characters of the input (comments)
- Efficient
    - Not XML, but ATerms
    - Maximal sharing
- Versatile
    - Extensible in three ways
    - Applied in many contexts

Implementation

- Open source (LGPL)
- C and Java API's
- (partially) doxygen/javadoc documented
- Available tools:
  - Binary/textual (maximally shared) (de)serialization
  - Syntax highlighting editors
  - Source code extraction
  - Pretty printers
  - Position annotation
  - Parse tree visualization
  - Ambiguity diagnostics
  - Rewriters
- http://www.meta-environment.org

Discussion

- Is there consensus on the I/O of parsing?
- {W,c,sh}ould your parser output UPTR as an alternative?
- Ideas, tips, improvements?