Implementing JITTs (Just-In-Time-Trees)

XML 2002

Patrick Durusau, pdurusau@emory.edu
Matthew Brook O'Donnell, matt@opentext.org

Background

- Complex texts (or analysis)
  - Multiple possible hierarchies
  - Only get one
  - Workarounds of varying utility
    - Inline: milestones, segmentation
    - Out-of-Line: standoff
    - Variant syntaxes: Mecs, TexMecs, LMNL

Variant Parsing

- BUVH: Bottom-Up Virtual Hierarchies
  - Parse forest of hierarchies
- Requires pre-parsing of texts
- Verbose markup
- Represents all possible trees
- But, are trees the problem?

Trees are a Symptom

- What composes a markup tree?
  - Quick answer: elements/PCDATA
- How to declare a markup tree?
  - Quick answer: DTD or schema
- But, what is a markup tree really?
  - A language, based on a meta-language
- And how are languages defined?

Defining A Language

- Standard Language Definition
  - –A language L is a set of strings over an alphabet.
  - –SGML/XML conflates lexical level with parsing of language
  - –Results in monolingual parsers

Monolingual Languages

- Language definition requires
  - Recognition of all tokens in the language
  - Recognition of what is not in the language
  - By its very nature, the definition of a language excludes others
  - Separate from parsing?

JITTs Reply – Yes!

- What is needed?
  - Definition of lexical level
  - Definition of parsing level
    - (separately)
- In XML
  - Lexical and parsing defined together
  - DTD/Schema defines a particular lexical vocabulary
  - Parsing is predefined
  - XML parser != JITTs parser

XML Parsers and JITTs

- Change error handling (XML Europe)
- Not true (parsing + lexical)
- Can change filtering
- Can’t alter conflation of lexical and parsing rules
- Filtering has several advantages
  - Multiple views of a single document
  - Validity or well-formedness of choice

Building a JITTs Parser

- JITTs parser requires
  - Definition of lexical level
  - Definition of parsing rules
    - (separately)
- SGML/XML documents
  - No changes required
  - Enhanced use of existing documents

Building a JITTs Parser II

- Don’t build from scratch!
  - Island Grammars!
- Specific productions match constructs of interest, “islands”
- General productions match the “water” around the island
- Separates the lexical from parsing

JITTs Advantages

- Tree based Access : SAX-like speed
  - JITTs simulation of DOM-Lite
    - Identical files, except:
      - 1 has markup
      - 2 has replacements that simulate JITTs parsing
- Markup Advantages without “<“, “>”
  - Return of Datatag (cf. Simon St. Laurent)
  - Processing like markup trees

Reasons for JITTs

- Single tree view of texts
  Vs.
- Multi-tree view of texts
- Dom-Lite
- Unlimited by current parser models
- Consider our Tree Friends
  - Simple tree or Complex tree
- Question is: Which do you prefer?
- Better Question: Which would your customers prefer?

Support for Research

- Support organizations that make this research possible!
  - Society of Biblical Literature: http://www.sbl-site.org
  - OpenText.org: http://www.opentext.org
  - TEI: http://www.tei-c.org

Island Grammar References

- Generating Robust Parsers using Island Grammars, Moonen, http://www.cwi.nl/~leon/papers/wcre2001/wcre200
- Lightweight Impact Analysis using Island Grammars, Moonen, http://citeseer.nj.nec.com/moonen02lightweight.ht
- Disambiguation Filters for Scannerless Generalized LR Parsers, Visser, www.cs.uu.nl/people/visser/ftp/BSVV02.pdf