CHALMERS | GÖTEBORG UNIVERSITY

Three Tools for Language Processing: BNF Converter, Functional Morphology, and Extract

Markus Forsberg

Thesis to be defended in public at **10:15, Sept. 25, 2007**
in EE, EDIT building, Göteborg
for the Degree of Doctor of Engineering.

The defense will be held in English.

Opponent: Directeur de Recherche de Classe Exceptionnelle
Gérard Huet, INRIA Rocquencourt, France

Department of Computer Science and Engineering
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Telephone +46-31-772 10 00

Abstract

Purely functional programming and meta programming based on declarative models are productive approaches to language processing and language resource building. Three tools are presented as evidence of this: *BNF Converter*, *Functional Morphology*, and *Extract*.

BNF Converter is a multi-lingual compiler tool. BNFC accepts as its input a grammar written in Labelled BNF (LBNF) notation, and generates a compiler front end: an abstract syntax, a lexer, and a parser. Furthermore, it generates a case skeleton usable as the starting point of back end construction, a pretty printer, a test bench, and a LaTeX document usable as a language specification. The program components can be generated in Haskell, Java, C, C++, Objective Caml, and C#, and their standard parser and lexer tools.

Functional Morphology and Extract are tools for creating lexical resources. Lexical resources, i.e. systematic computational descriptions of words in a natural language, are fundamental resources for any language technology application and it is imperative that they are of high quality. Moreover, since the development of lexical resources is such a time-consuming task, it is important that they can be created efficiently. The tools have been created to address these issues.

Functional Morphology (FM) is a Haskell library for defining lexical resources. A lexical resource in FM is defined using the *word-and-paradigm* model. Paradigms are abstractions of inflection tables, represented as functions over hereditarily finite algebraic data types, and the lexicon consists of a list of words in citation form annotated with paradigm identifiers. The runtime system of FM consists of an inflection engine, an analyzer, a synthesizer and a compiler to many standard lexicon formats.

Extract is a special-purpose tool for extracting annotated words from raw text data. The extraction is based on a set of rules, where a rule is a propositional formula where the atoms of the formula are regular expressions. The regular expressions, corresponding to a subset of the word forms in a paradigm, contain variables used for capturing substrings. A recent addition to Extract is *Constraint Grammar* constructs together with the possibility of a structured input format. This addition enables a rule to refer to the contexts of the word forms and additional information added to them, such as part of speech (POS) tags.