Type-checking in Formally non-typed Systems

Dick Crouch
Powerset, Inc.
San Francisco, USA
crouch@powerset.com

Tracy Holloway King
Palo Alto Research Center
Palo Alto, USA
thking@parc.com

Abstract

Type checking defines and constrains system output and intermediate representations. We report on the advantages of introducing multiple levels of type checking in deep parsing systems, even with untyped formalisms.

1 Introduction

Some formalisms have type checking as an inherent part of their theory (Copestake (2002)). However, many formalisms do not require type checking. We report on our experiences with a broad-coverage system for mapping English text into semantic representations for search applications. This system uses the XLE LFG parser for converting from text to syntactic structures and the XLE ordered-rewriting system to convert from syntax to semantic structures. Neither component formally requires type checking. However, type checking was introduced into the syntactic parser and at multiple levels in the semantics in response to the engineering requirements on a large-scale, multi-developer, multi-site system.

2 Syntactic Typing

The syntactic parser outputs a tree and an attribute value matrix (f(unctional)-structure). Meaning-sensitive applications use the f-structure which contains predicate argument relations and other semantically relevant dependencies.

A feature declaration (FD) requires every f-structure attribute to be declared with its possible values. These values are typed as to whether they are atomic or are embedded f-structures. (1) shows the FD for NUM(ber) and SPEC(ifier). NUM takes an atomic value, while SPEC takes an f-structure containing the features ADJUNCT, AQUANT, etc.

(1) a. NUM: -> $ {pl sg}.
b. SPEC: -> << [ADJUNCT AQUANT DET NUMBER POSS QUANT SPEC-TYPE].

XLE supports overlay grammars where a grammar for an application uses another grammar as its base. The FDs form part of the overlay system. For example, there is an FD used by the Parallel Grammar project (Butt et al. (2003)); the standard English FD adds and modifies features; then domain specific FDs overlay this. (2) gives the number of features in the ParGram FD and the standard English overlay.

(2)

	atomic	f-structure
English	76	33
ParGram	34	11

The grammar cannot be loaded if there is a feature or value that is not licensed by the FD (to type check the lexicon, the generator is loaded). The command `print-unused-feature-declarations` can be used after a large parse run to determine which features never surfaced in the analysis of the corpus and hence might be candidates to be removed from the grammar.

As LFG does not have type checking as part of its theory (Dalrymple et al. (2004)), XLE originally did not implement it. However, in grammar engineering, type checking over features speeds up the development process and informs later processes and applications what features to expect since the FD serves as an overview of the output of the grammar.

Software Engineering, Testing, and Quality Assurance for Natural Language Processing, pages 3–4,
Columbus, Ohio, USA, June 2008.

3 Semantic Typing

The syntactic output is the input to several sets of ordered rewriting rules that produce semantic structures (Crouch and King (2006)). The nature of ordered rewriting systems, which consume input facts to create novel output facts, makes type checking extremely important for determining well formedness. When these representations are used in applications, type declarations can document changes so that the subsequent processing can take them into account.

The semantic typing is done by declaring every fact that can appear in the structure, its arity, and the type of its arguments. A field is available for comments and examples. (3) shows the licensing of nominal modifiers in noun-noun compounds (nn_element), where *skolem* and *integer* are argument types.

(3)
```
|- type(proposition,
   nn_element(%%Element:skolem,
              %%Head:skolem,
              %%Nth:integer),
   comment(["%%Element is the %%Nth
   term in the compound noun %%Head
   Example {NP: the hinge oil bottle}
   in_context(t,nn_element(hinge:10,bottle:1,2))"])).
```

The xfr semantics is developed by multiple users. By breaking the rules into modules, type checking can occur at several stages in the processing pipeline. The current system provides for type checking at word-prime semantics, the final semantics, and abstract knowledge representation. (4) shows the number of (sub)features licensed at each level.[1]

(4)

word prime	91
lexical semantics	102
akr	45

In addition to aiding the developers of the semantics rules, the type declarations serve as documentation for the next steps in the process, e.g. creating the semantic search index and query reformulation.

4 Additional Engineering Support

The semantic type checking is a set of ordered rewrite rules, using the same mechanism as the semantics rules. As such, the notation and application are familiar to the grammar engineers and hence more accessible. Since the type checking involves additional processing time, it is not part of run-time processing. Instead, it is run within a larger regression testing regime (Chatzichrisafis et al. (2007)). Grammar engineers run a core set of regression tests before checking in any changes to the svn repository. Larger nightly runs check performance as well as typing at all levels of analysis and help ensure compatibility of changes from multiple developers.

The syntactic grammar cannot be loaded with feature type violations. However, the nature of an ordered rewriting system makes it so that loading the rules does not give the full feature type space of the resulting output. To force compliance with type checking requirements, check-ins require regression tests before committing changes. The output of these tests is type checked and, if unlicensed features are found, the commit is blocked. The grammar engineer can then update the type checking rules or modify the semantic rules to produce only licensed features. The regression testing is then rerun and, if the type checking passes, the commit proceeds.

In sum, introducing type checking at multiple levels provides a better development environment for grammar engineers as well as documentation for the developers and for applications.

[1] A stripped-down XML version of the semantics uses an xschema which checks that only the reduced feature set is used and that the XML is well-formed.

References

Butt, M., Forst, M., King, T.H. and Kuhn, J. 2003. The Feature Space in Parallel Grammar Writing. In *ESSLLI Workshop on Ideas and Strategies for Multilingual Grammar Development.*

Chatzichrisafis, N., Crouch, D., King, T.H., Nairn, R., Rayner, M. and Santaholma, M. 2007. Regression Testing for Grammar-based Systems. In *Grammar Engineering Across Frameworks.*

Copestake, A. 2002. *Implementing Typed Feature Structure Grammars.* CSLI.

Crouch, D. and King, T.H. 2006. Semantics via F-Structure Rewriting. In *Proceedings of LFG06.*

Dalrymple, M., Kaplan, R. and King, T.H. 2004. Linguistic Generalizations over Descriptions. In *Proceedings of LFG04.*