Modelling Robustness with Conjunctive Grammars
Motivation

I DO WHAT I WANT
Motivation

• No ideal model for robust parsers
• Island parsers are often idiosyncratic
• Have fun with conjunctive grammars
• Grammars in da cloud
• ...
• PROFIT?
What's a grammar?

- Language definition
  - characteristic function of the language
  - iterator for language elements
- “grammatical correctness”
- Commitment to grammatical structure
- $<N,T,P,s,...>$

URGH! THRAK CREATE FIRE! FIRE HOT!

FIRE NEAT, THRAK, BUT ME INVENT CLUB! CLUB FOR HIT THINGS WITH.

OOH...

HELLO, FRIENDS! IT SEEMS THAT I HAVE INVENTED GRAMMAR.

THANKS! ME AND THRAK CAN COMMUNICATE FAR MORE EASILY NOW THAT WE HAVE HELPING VERBS AND ARTICLE ADJECTIVES.

"THRANK AND I!"

Cyanide and Happiness © Explosm.net
What about semantics?

• Grammars define syntax
• Syntax is just the beginning of semantics
• ...or is it?
• Colorless green ideas sleep furiously.

Noam Chomsky
Despite the undeniable interest and importance of semantic and statistical studies of language, they appear to have no direct relevance to the problem of determining or characterizing the set of grammatical utterances. I think that we are forced to conclude that grammar is autonomous and independent of meaning.
Grammars define structure
&
can assume
different semantics
What’s a conjunctive grammar?

- Classic grammars define **sets** of words
- Set operations:
  - disjunction / choice / addition
  - conjunction / intersection
  - negation
- Purely theoretical extension
- **Scarce** Some practical uses

What’s robustness?

Figure 10. A spectrum of approaches for source code analysis.

- Tolerance towards language dialects
- Agile grammar hacking engineering
- Negotiated transformations
- Information recovery heuristics

S. Klusener, R. Lämmel, *Deriving Tolerant Grammars from a Base-line Grammar*, ICSM 2003
Island grammars

- Detailed production rules for interesting constructs
- Liberal production rules for the rest
  - ~[\.]+ [\.] → Statement
  - ~[\ \t\n]+ → Water {avoid}
- Minimal set of assumptions about the overall structure
  - (e.g., “a program is a list of statements”)

Conjunctive clauses

• Statement is a chunk between dots/semicolons/…
• Statement is *also* something else
  • keyword, expression, block
• So, we *define* a statement
  • as an “island” *and* as a statement
Assumed semantics 1

• Take a conjunctive robust grammar
• Parse classically as a conjunctive grammar
  • recursive descent or generalised LL
• Run over a sufficiently big reference codebase

• $\Rightarrow$ validation of the robust grammar
Assumed semantics 2

• Take a conjunctive robust grammar
• Parse only with detailed clauses
• If failed, backtrack to tolerant clauses
  • locally

• \(\Rightarrow\) disciplined error recovery
Assumed semantics 3

- Take a conjunctive robust grammar

- Parse only with tolerant clauses
  - obtain the global structure

- Parse the islands with subgrammars
  - if possible

- ⇒ grammarware as a service
Parsing in the cloud

Baseline grammar

Adapted grammar

grammar transformation

compilation-unit:
    using-directives? global-attributes? namespace-member-declarations?

using-directives:
    using-directive

using-directive:
    using-alias-directive
    using-namespace-directive

using-alias-directive:
    "using" id "=" namespace-or-type-name ";

...
Parsing in the cloud

Baseline grammar

Adapted grammar

vertical(using-directives);
deyaccify(using-directives);
inline(using-directives);
inline(using-alias-directive);
inline(using-namespace-directive);
massage(using-directive+?, using-directive*);
factor(
("using" identifier "=" namespace-or-type-name ";") | ("using" namespace-name ";")
| "using" (identifier "=" namespace-or-type-name) | (namespace-name) ";"));

...
Parsing in the cloud

Baseline grammar

Adapted grammar

grammar transformation

compilation-unit:
  ("using" using-directive-insides ";")* ("[" "assembly" ":"
  ga-section-insides "]")* namespace-member-declaration*

namespace-member-main:

  "namespace" qualified-identifier class-base?
  
  "{" namespace-body-insides "}" ";"?

  "class" identifier class-base?

  "{" class-member-declarations? "}" ";"?

  "struct" identifier struct-interfaces?

  "{" struct-member-declarations? "}" ";"?

  "interface" identifier interface-base?

  "{" interface-member-declarations? "}" ";"?

  "enum" identifier enum-base? 
  "{" enum-body-insides "}" ";"?

  "delegate" type id "{" formal-parameter-list? "}" ";";

...
layout $L = \left[ \begin{array}{c} \ \t\r\n \end{array} \right]^{*} !>> \left[ \begin{array}{c} \ \t\r\n \end{array} \right] ;$

syntax CompilationUnit = ("using" NotSemicolon ";")*
("[" "assembly" ";"")* NotRightSquareBracket "]")* NamespaceMemberDeclaration* ;

syntax NotRightSquareBracket = NRSBChunk+ () >> [\]];
lexical NRSBChunk = ![\]\t\r\n]+ >> [\]\t\r\n];
Parsing in the cloud

Using-directive-insides:

Ga-section-insides:

Attribute-section-insides:

Baseline grammar

Grammar transformation

Adapted grammar

Grammar transformation + subgrammar

Subgrammar

Skeleton grammar

Delivering islands

Island subgrammars

Parsing the overall structure

Parsing the islands

Parse tree skeleton

Refined parse tree

Filling the gaps
Related work

- Quasi-synchronous grammars
  - natural language translation framework
- Parallel parsing
  - usually non-distributed, but concurrent
- Ambiguity elimination
  - ambiguity is bad, okay?
- Permissive grammars
  - explicit error recovery rules
Related work

• Quasi-synchronous grammars

• Parallel parsing

• Ambiguity elimination

• Permissive grammars
Current/future work

- Robustness techniques & tolerance spectrum
- Semi-parsing with Boolean grammars
- Validation/testing of skeleton grammars
Stay tuned!

vadim@grammarware.net