SWAT

Modelling Robustness with Conjunctive Grammars

POWERED BY GRAMMARLAB

6th SATToSE 2013, Bern
Vadim Zaytsev, SWAT, CWI

Motivation

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,…>*

P. Klint, R. Lämmel, C. Verhoef. *Toward an Engineering Discipline for Grammarware*. ToSEM, 2005.

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.

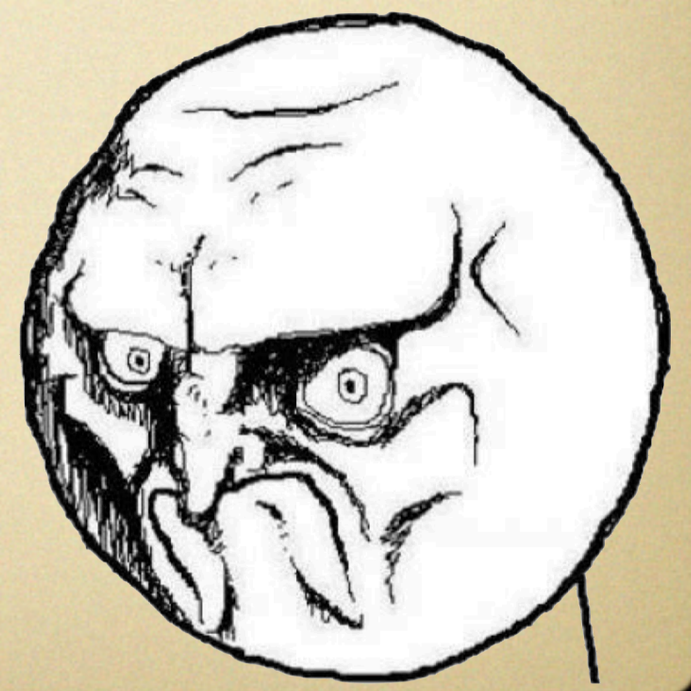
THANKS! ME AND THRAK CAN COMMUNICATE FAR MORE EASILY NOW THAT WE HAVE HELPING VERBS AND ARTICLE ADJECTIVES.

"THRAK AND I"

What about semantics?

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

Noam Chomsky →

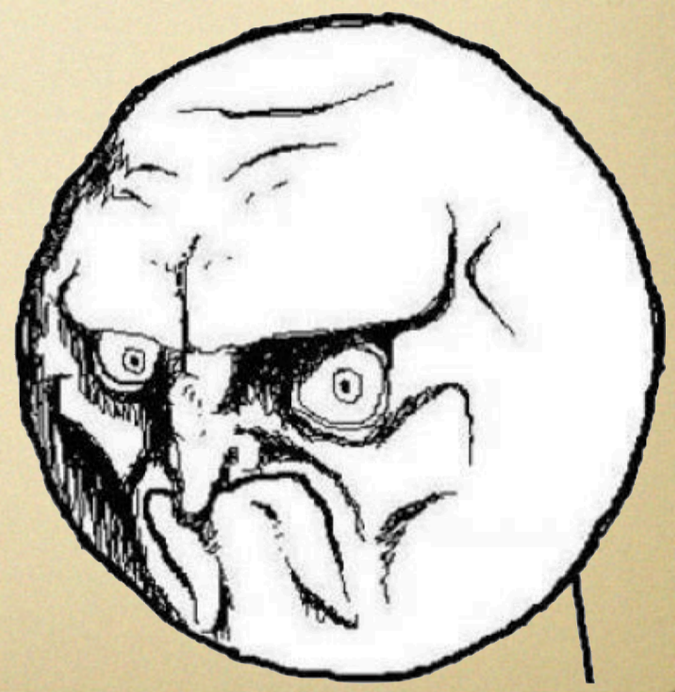


Syntactic Structures

(1957)

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.

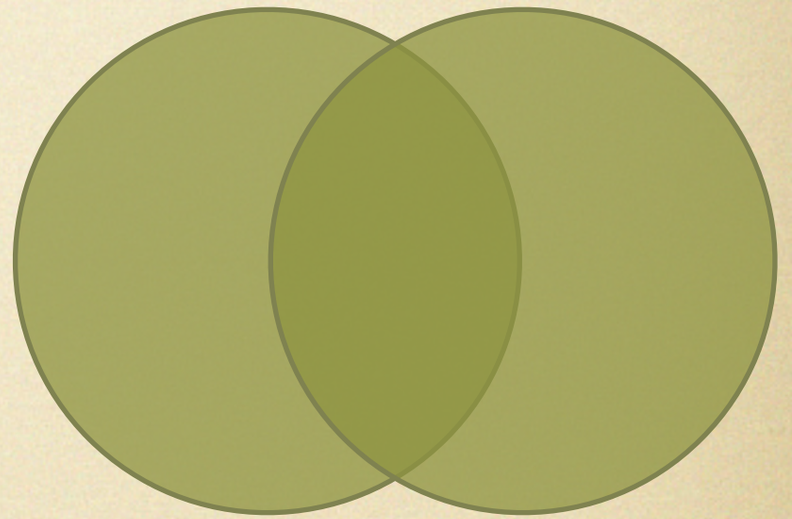
Noam Chomsky →



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



A. Okhotin. *Conjunctive Grammars*. JALC, 2001.

What's robustness?

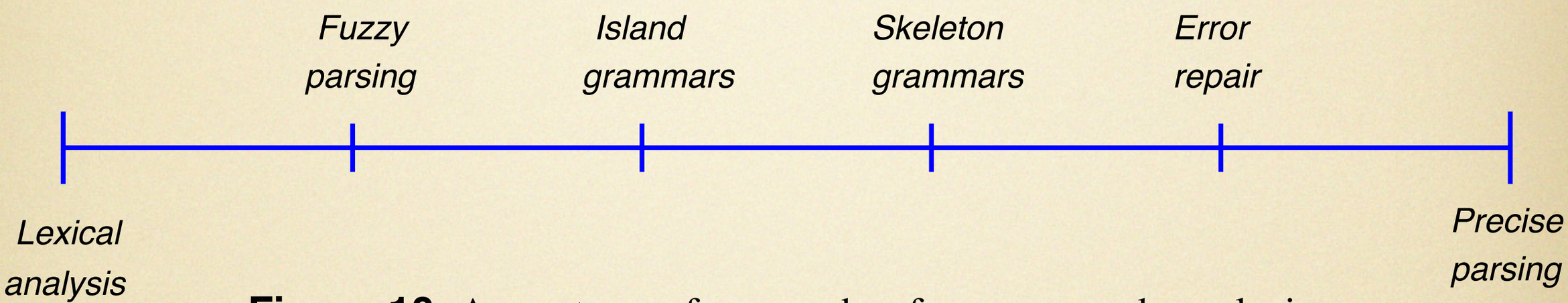


Figure 10. A spectrum of approaches for source code analysis.

- 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

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”)

A. van Deursen, T. Kuipers, *Building Documentation Generators*, ICSM 1999.
L. Moonen, * *using Island Grammars*, WCRE 2001, IWPC 2002.

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
- ⇒ validation of the robust grammar

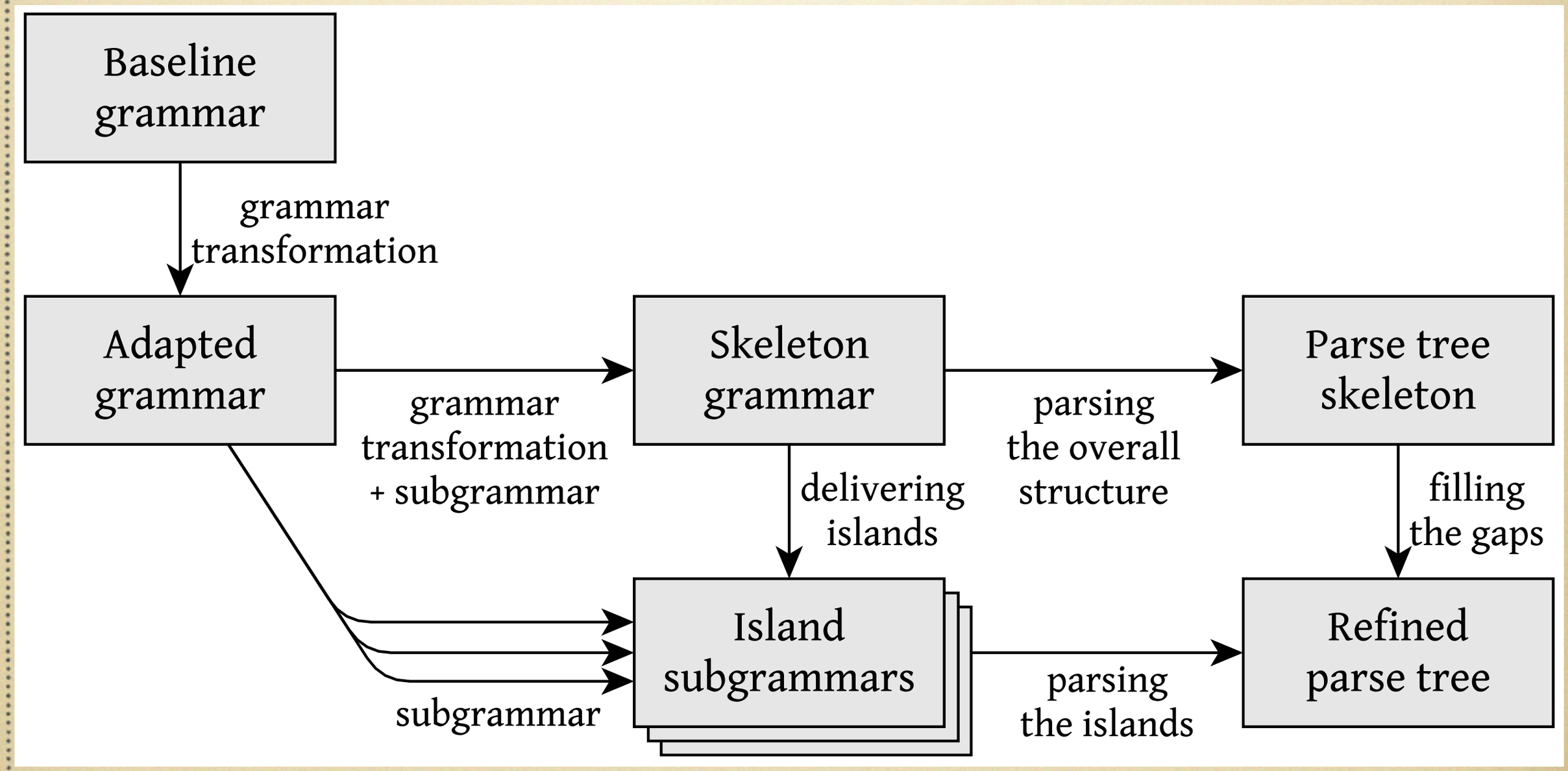
Assumed semantics 2

- Take a conjunctive robust grammar
- Parse only with detailed clauses
- If failed, backtrack to tolerant clauses
  - locally
- ⇒ disciplined error recovery

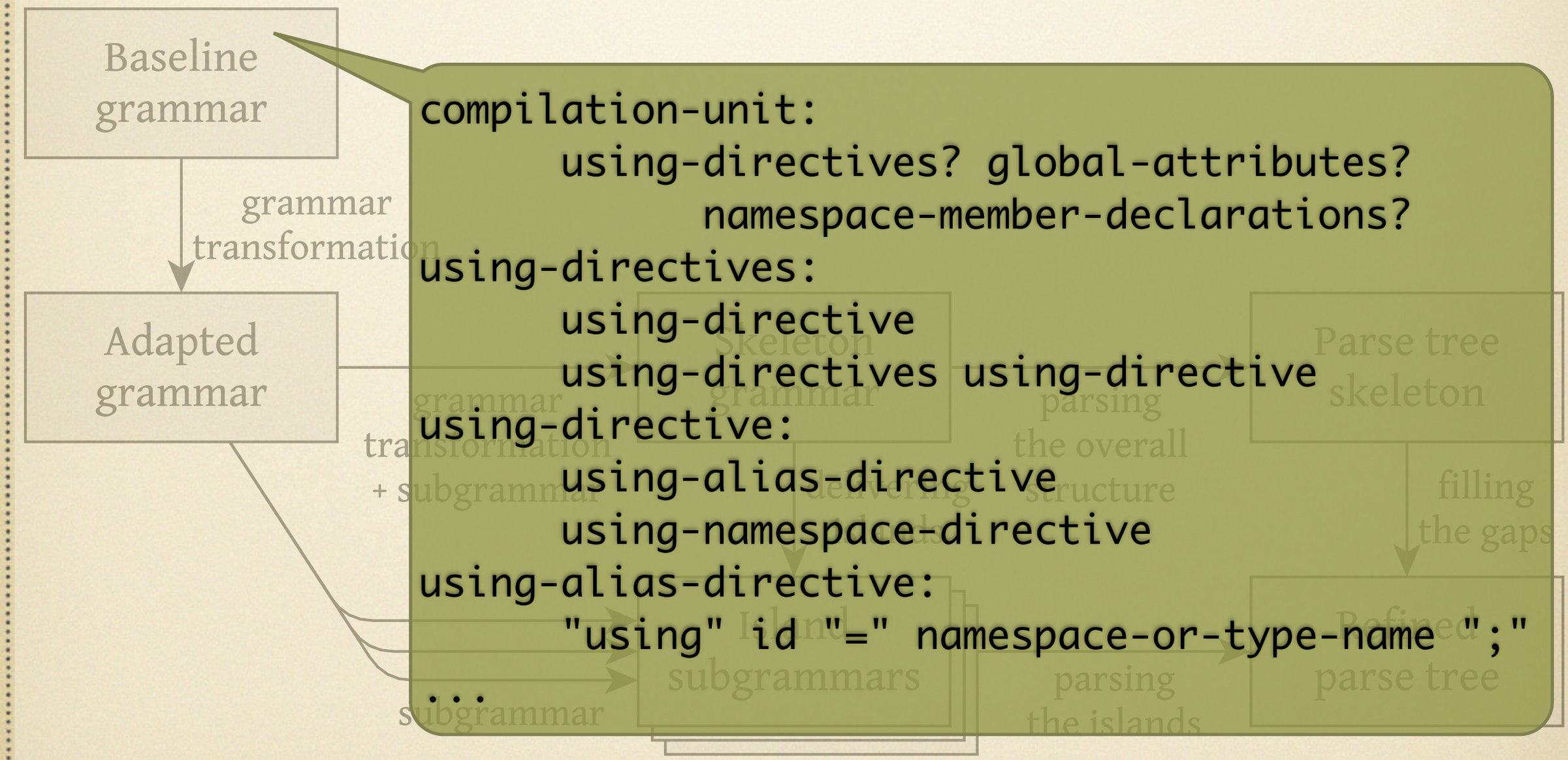
Assumed semantics 3

- 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

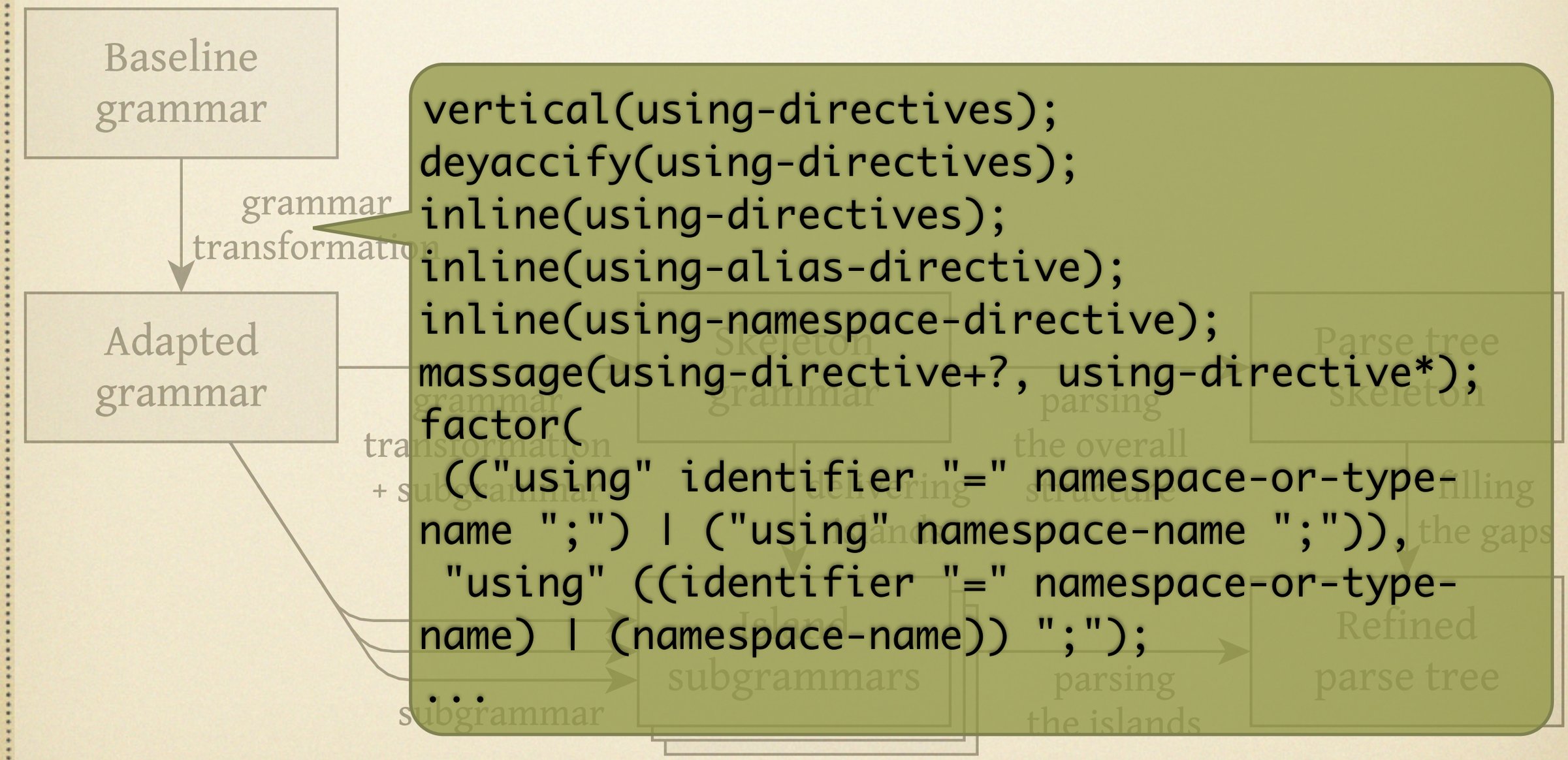
Parsing in the cloud



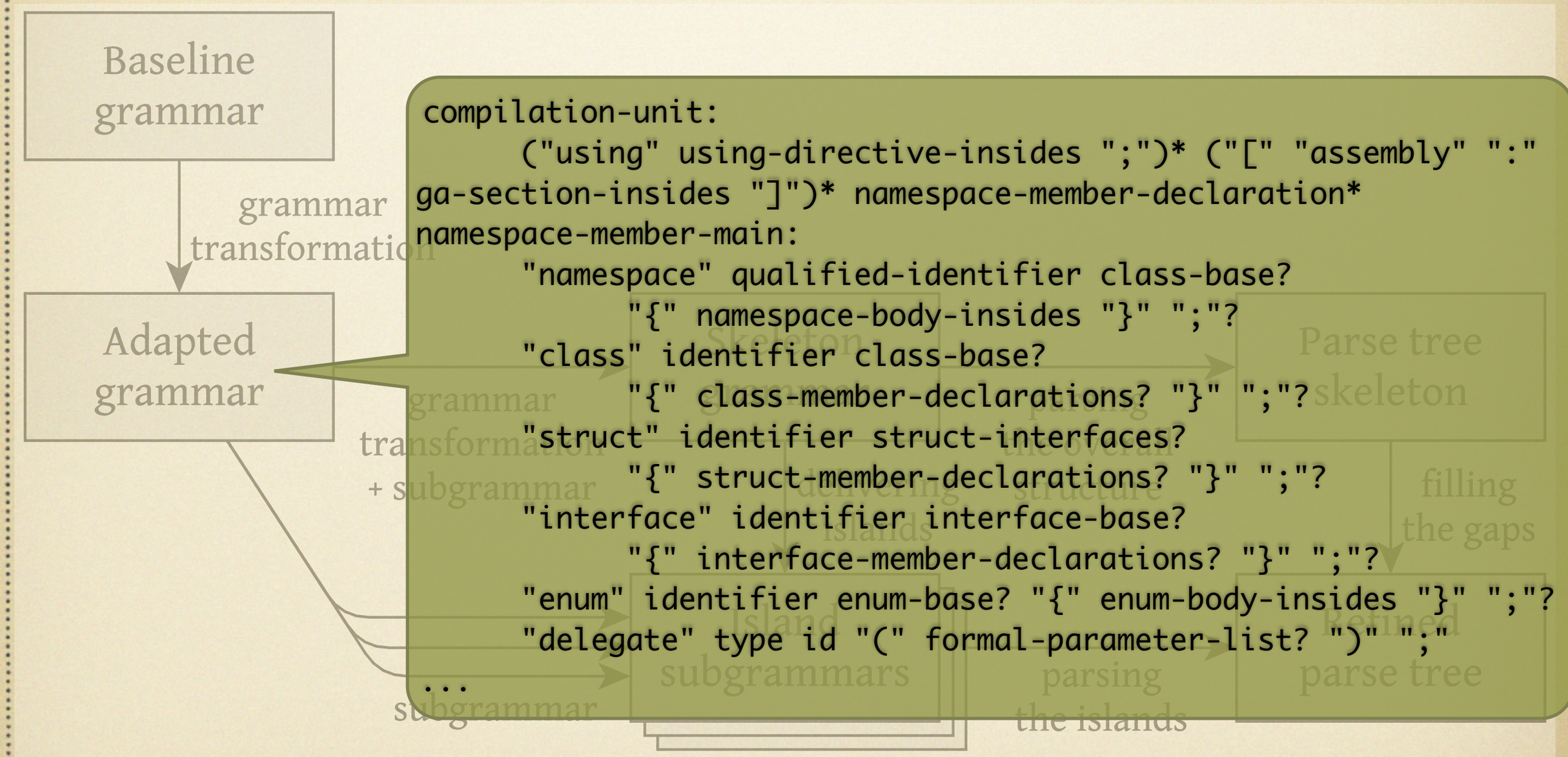
Parsing in the cloud



Parsing in the cloud



Parsing in the cloud

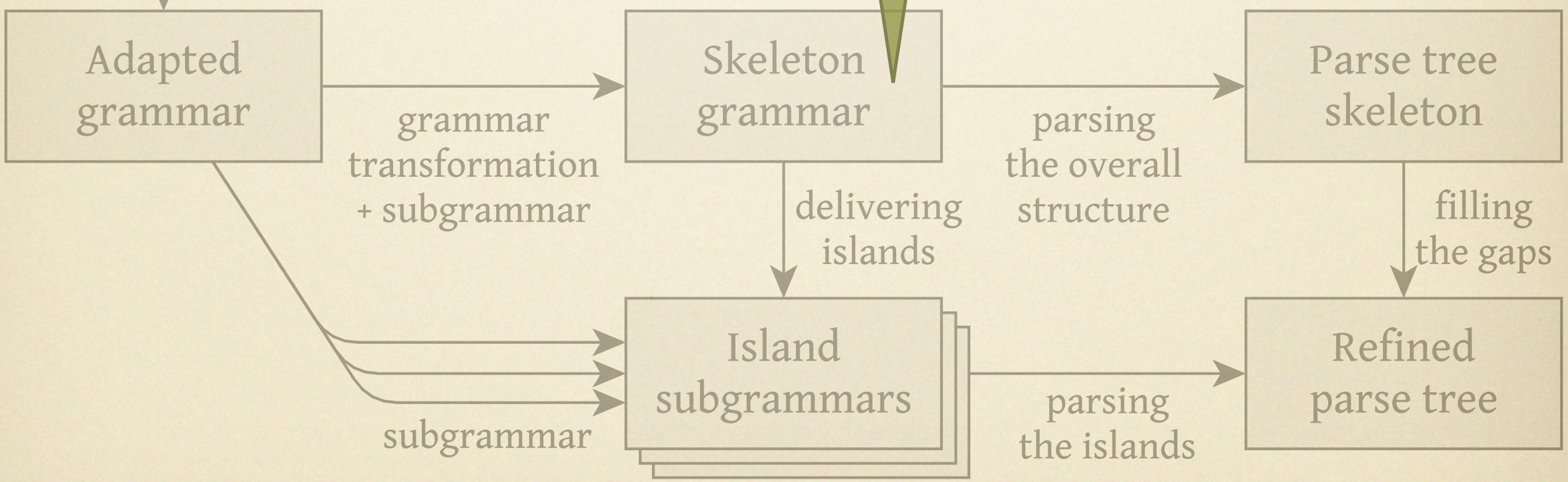


```
layout L = [\ \t\r\n]* !>> [\ \t\r\n] ;

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

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

...
```

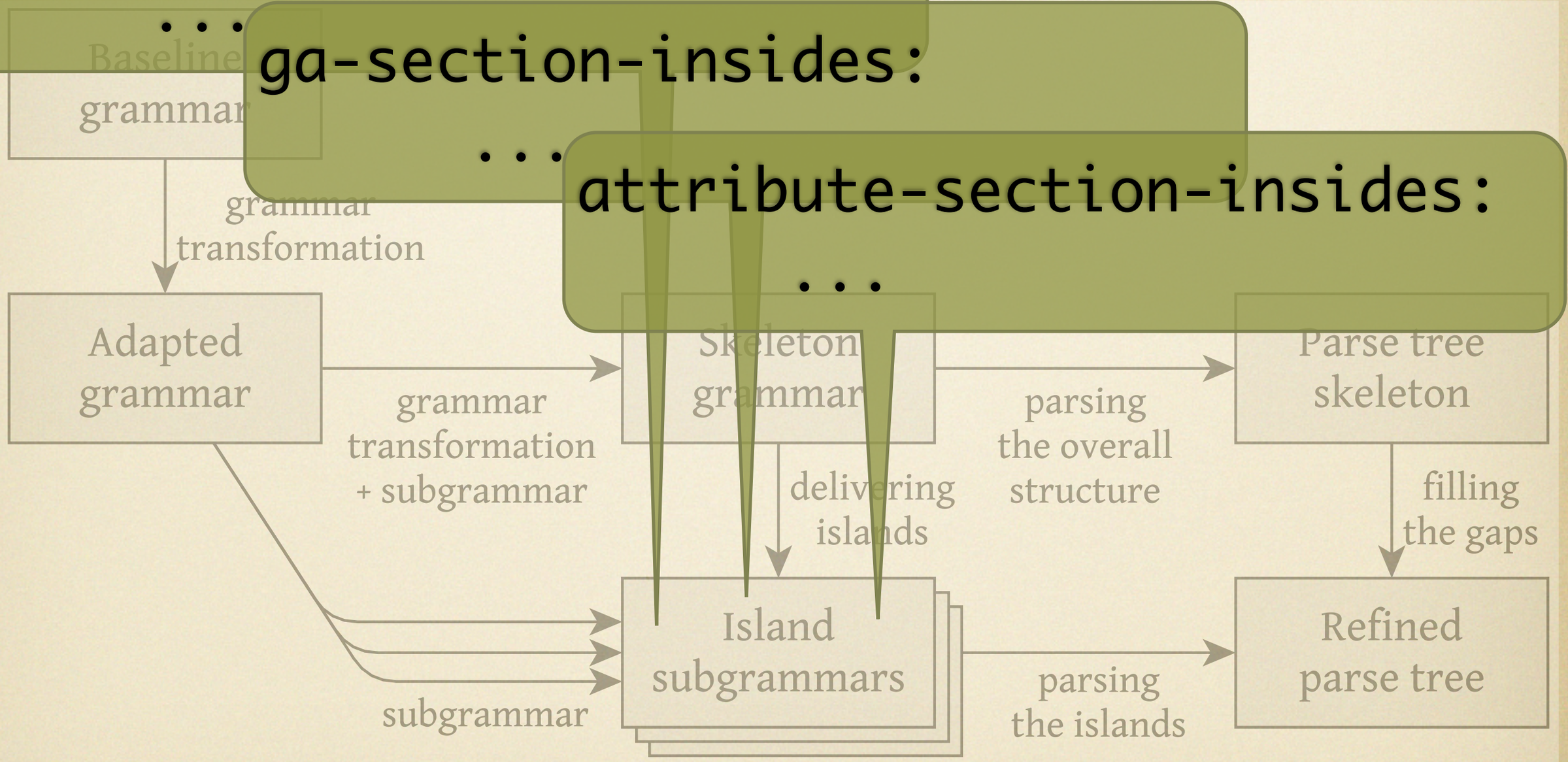


Parsing in the cloud

```
using-directive-insides:
...
```

```
ga-section-insides:
...
```

```
attribute-section-insides:
...
```



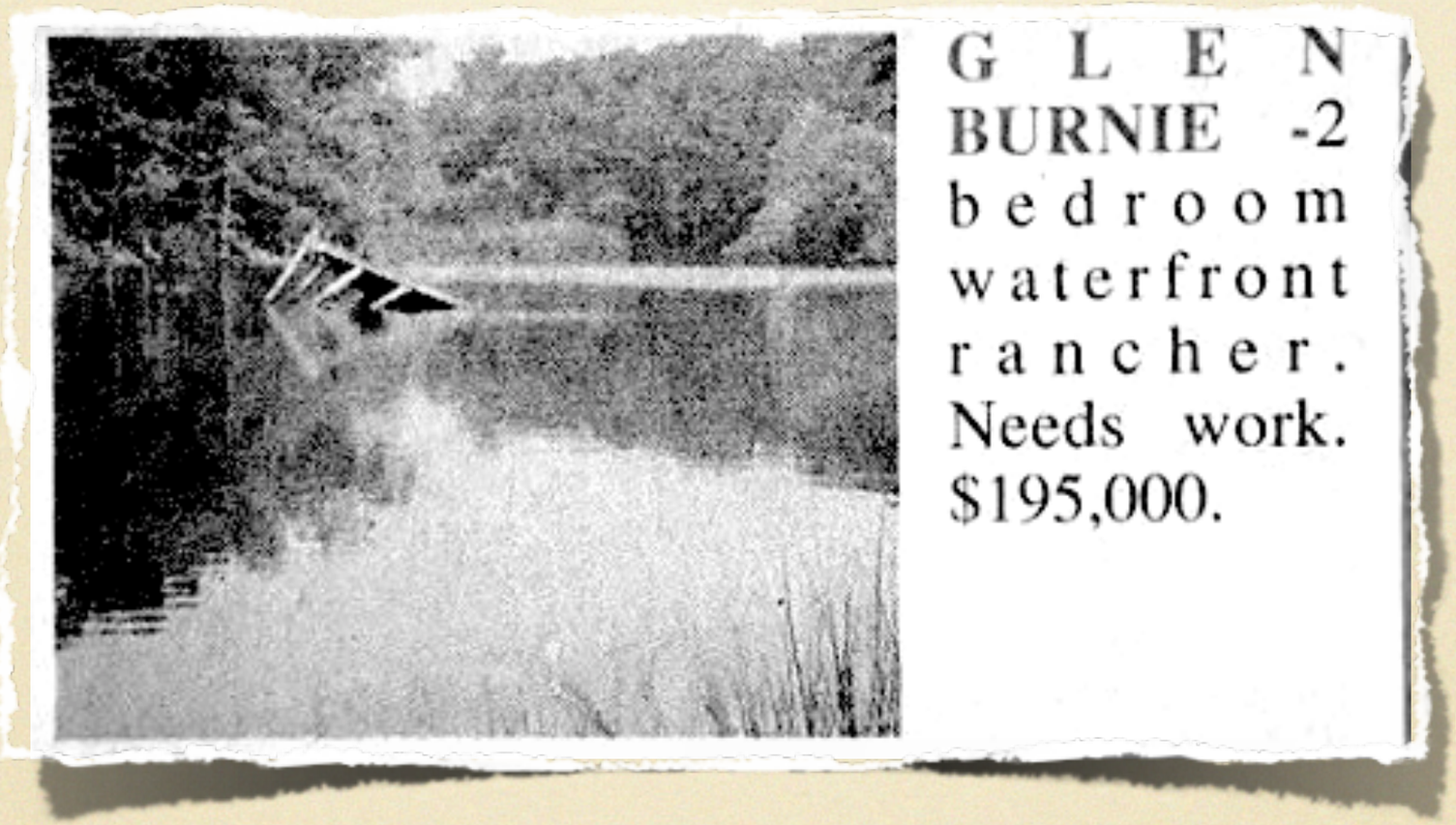
Related work

- 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
  - D. A. Smith, J. Eisner. *Quasi-Synchronous Grammars: Alignment by Soft Projection of Syntactic Dependencies*. StatMT 2006.
- Parallel parsing
  - H. Alblas, R. op den Akker, P. O. Luttighuis, K. Sikkel. *A Bibliography on Parallel Parsing*. SIGPLAN Notices 1994.
- Ambiguity elimination
  - H. J. S. Basten, *Tracking Down the Origins of Ambiguity in Context-free Grammars*, ICTAC 2010.
- Permissive grammars
  - L. C. L. Kats, M. de Jonge, E. Nilsson-Nyman, E. Visser. *Providing Rapid Feedback in Generated Modular Language Environments. Adding Error Recovery to SGLR Parsing*. OOPSLA 2009.

Current/future work



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

POWERED BY

GRAMMARLAB

Stay tuned!

vadim@grammarware.net