Modelling Robustness with Conjunctive Grammars  
(work under development)

Vadim Zaytsev, vadim@grammarware.net  
Software Analysis & Transformation Team (SWAT),  
Centrum Wiskunde & Informatica (CWI), The Netherlands

Grammars in a broad sense [2] are used to specify structural commitment 
in software systems: concrete syntax definitions determine how source code is 
turned into a parse tree; library interfaces define the signatures of the functions 
exposed and used by third parties; document schemata fix the structure of the 
XML documents that are considered valid; type definitions influence assertions 
on the variable values and function bodies; etc. In the context of software evo-
lution, one frequently faces a challenge of consistency enforcement and change 
propagation: when the source code is updated, its structural commitments must 
be reevaluated — commonly by coevolving a corresponding grammar.

There exist numerous techniques to address robustness issues: tolerant [4], ag-
formation; notation-parametric [9] heuristics of grammar recovery; and many 
others. These techniques can be considered to share one important property — 
they specify two kinds of structural commitment at the same time: a precise one 
and a tolerant one. On one extreme, a precise commitment relation is unknown 
or missing, so tolerance is the only way to ensure robustness. On the other ex-
treme, a tolerant kind of commitment is never needed. In this presentation, it 
is proposed to formally specify such double commitments with conjunction [6] — 
an operation commonly found in set theory, but much more rare in grammarware. This approach may be seen as distantly related to quasi-synchronous 

By using a conjunctive grammar to specify both precise and tolerant struc-
tural commitments, we create a setup where one entity specifies many ways of 
jeopardising or weakening of existing contracts when the base software evolves. 
For instance, when parsing, any failure of a conjunctive clause of a particular 
nonterminal can be noted and reported, but does not necessarily prevent deliver-
ing a parse tree to the next tool in the pipeline (such as a fact extractor).

During the presentation, there will be a demonstration showing conjunctive 
grammars to be useful for specifying derived grammars in the islands-and-lakes 
paradigm, which is a fuzzy generalisation of context-free grammars that allows 
insignificant fragments of “water” to be recognised along the detailed “islands” 
for the sake of performance or robustness. We will use Rascal [3] language work-
bench, and all the code will be made publicly available through the Software 
Language Processing Suite repository [10].
References


---

1 The authors are given according to the list of contributors at [http://github.com/grammarware/slps/graphs/contributors](http://github.com/grammarware/slps/graphs/contributors).