# Context-Sensitive Parsing Through Stateful Parsing

Nicolas Laurent Kim Mens
Université catholique de Louvain, ICTEAM
{nicolas.laurent, kim.mens}@uclouvain.be

## Problem

Context-sensitive parsing remains a problematic area within parsing research. Many mainstream programming and markup languages (C, C++, Haskell, Python, XML, ...) possess context-sensitive features. These features are traditionally handled with ad-hoc code (e.g., custom lexers).

Some approaches (data-dependent grammars, monadic parsing, attribute grammars, definite clause grammars, ...) are able to express context-sensitive features. However, none of these approaches, with the exception of backtracking semantic actions [2], feature context-transparent constructs (grammar rules, parsers in a parser-combinator approach, ...).

A grammatical construct is context-transparent if it is unaware of the context shared between its ancestors and its descendants. Consider three non-terminals A, B and C such that A's right-hand side contains B and B's right-hand side contains C. If A and C must share some data, a context-transparent system will not require B to ferry over the data between A and C, nor to know anything about the data.

This lack of context transparency makes grammars hard to write, maintain and compose, as each change to a construct may require changing each construct through which it is accessed.

## Solution

To enable context-sensitive yet context-transparent parsing, we propose *principled stateful parsing* [3] as a new transactional discipline that makes state changes transparent to parsing mechanisms such as backtracking and memoization.

Our approach is based on PEGs [1], a formalization of top-down recursive-descent parsers. The model is well-suited to stateful execution, as parser execution order is deterministic and predictible.

In our approach, each parser is able to access a mutable data store and use it to make parsing decisions or communicate with other parsers. Whenever a parser fails to match, the changes it induced on the data store need to be undone. This is achieved by taking a **snapshot** of the state prior to the parser's execution, and **restoring** it as needed. This leads to a very simple contract that each parser must respect: either it succeeds; or it fails, undoing any state changes it induced in the process.

It is sometimes desirable to save the result of a parser's excecution, i.e., the state changes it induced, by performnig a **diff** between the states before and after the execution. The natural inverse is the ability to **merge** these changes back into the state. The most straightforward application of *diff* and *merge* capabilities is the memoization of parse results. However, other valuable use cases exist, such as longest-match parsing and left-recursive PEG parsing.

A system outfitted with these four primitive operations (**snapshot**, **restore**, **diff** and **merge**) whose parsers respect the *suceed-or-undo* contract enables convenient definition of context-sensitive language features through safe stateful parsing. More details on our approch and the Autumn library implementing it can be found in our SLE paper [3].

## Challenges and Opportunities

The first challenge is efficient execution. Given that memory allocation is a slow operation, the *snapshot* operation quickly becomes a bottleneck. Furthermore, PEG parsers sometimes use memoization to improve their performance; however the introduction of state besides the input position means that the procedure to decide whether to recall a memoized result becomes much more expensive.

The second challenge is to alleviate the requirements on the user. As is, he must guarantee that each parser satisfies the *succeed-or-undo* contract, and supply data structures that support our four primitive operations. A rich library of parsers and data structures greatly decreases the pain, but is it enough?

We wonder whether a hybrid of our approach and that of Thurston and Cordy [2] couldn't solve some of these issues.

Regarding opportunities, it seems that recording intermediary snapshots could lead to fully general iterative parsing in the presence of context-sensitivity. Likewise, the availability of this information could greatly benefit grammar debugging.

## References

[1] B. Ford. Parsing expression grammars: A recognition-based syntactic foundation. *POPL 2004*.

[2] A. Thurston, J. Cordy. A backtracking LR algorithm for parsing ambiguous context-dependent languages. *CASCON 2006*.

[3] N. Laurent, K. Mens. Taming context-sensitive languages with principled stateful parsing. To appear at *SLE 2016*.