Relit: Implementing Typed Literal Macros in Reason

CHARLES CHAMBERLAIN, University of Chicago
CYRUS OMAR, University of Chicago

Reason is an increasingly popular alternative syntax for OCaml designed to make OCaml more syntactically familiar to contemporary programmers. However, both Reason and OCaml build in literal notation for only a select few data structures, e.g. lists, arrays and, in the case of Reason, a variant on HTML notation. This is unsatisfying — there are many other notations that are familiar to programmers in various domains where OCaml might be useful.

In a paper to appear at ICFP 2018, Omar and Aldrich address this problem by introducing *typed literal macros (TLMs)*. TLMs allow library providers to define new literal notation for the data structures that they have defined. Unlike other prior approaches, e.g. `camlp4` and `ppx`-based string rewriting, both explored in the OCaml ecosystem, TLMs come equipped with powerful abstract reasoning principles — clients do not need to "peek at" the underlying expansion or the implementation of the parser to reason about types and binding. The paper by Omar and Aldrich investigates these abstract reasoning principles in formal detail.

The purpose of this proposed talk is to provide additional details of our implementation of TLMs for Reason, which makes sophisticated use of the existing `ppx` system in OCaml together with an encoding technique based on singleton signatures to avoid needing to modify the OCaml compiler itself, despite the fact that TLMs integrate into the type and binding structure of the language. We make only a small number of conservative changes to the Reason grammar (and speculate on analagous changes that could be made to the base OCaml grammar to support TLMs in programs not written in Reason). We reflect on some of the challenges that we faced in interfacing with the various components of the OCaml system.

1 INTRODUCTION

Motivation similar to that in the ICFP paper – but compare more explicitly to how people use ppx to rewrite string literals to support notation, camlp4, and Reason's built in HTMLish notation.

2 OVERVIEW BY EXAMPLE

Cite the ICFP paper, give a simple example, list the reasoning principes

3 IMPLEMENTATION

(1) Extending the Reason parser
(2) Using singleton signatures to encode definitions
(3) Using exceptions for application + typechecking inside a PPX (but really we only need to signature check but thats not possible). Is it bad?
(4) Using ocamldep to determine module dependencies
(5) Awkwardness with packaging the parser
(6) Speculation: issues with OCaml + Reason support together (use menhir's parameterization?)

4 DISCUSSION

Successes:

(1) ocamldep was a straightforward way to do context independence
(2) can do typechecking and compile-time code execution in a ppx

Challenges:

(1) packaging / loading parsers at compile-time is a huge hassle

Authors' addresses: Charles Chamberlain, University of Chicago, cchamberlain@uchicago.edu; Cyrus Omar, University of Chicago, comar@cs.uchicago.edu.

(2) jbuilder treats ppx differently
(3) ordering of ppx matters
(4) ocamldep can't see into spliced terms – how to fix?
(5) we need to extend merlin and other tools still...
(6) issues with using OCaml + Reason together

briefly: relationship between this stuff and Scheme/Racket/Scala-style macros

REFERENCES