#5 Modular Type Constructors

Submitted 392kB 8 Jul 2014 2:58:09pm EDT | f4d351ee53566ea4fd0cfef48b4dde401c88d3ee

You are an **author** of this paper.

▶ ABSTRACT

Abstraction providers sometimes need to directly introduce new types and operators into existing languages. Ideally, such typed language fragments could be separately defined [more]

▶ AUTHORS

C. Omar, J. Aldrich [details]

▶ TOPICS AND OPTIONS

	OveMer	Con
Review #5A	C	Y
Review #5B	D	Z
Review #5C	D	Y
Review #5D	D	Y

Edit paper | Add response

Reviews in plain text

Review #5A

Modified 14 Aug 2014 10:59:14pm EDT Plain text

OVERALL MERIT (?)

C. Weak paper, though I will not fight strongly against it

CONFIDENCE (?)

Y. I am knowledgeable in this area, but not an expert

COMMENTS FOR THE AUTHORS

The paper introduces a calculus, @lambda, which aims at allowing the modular
introduction of and reasoning about new types and operators as an alternative
to creating new language dialects to integrate new features seamlessly into a
programming language.

The approach is based on several distinct languages: an external language,
which can be extended using a static language, which at the same time is the
type level language, and a fixed internal language to which external language
programs get mapped to via a bidirectionally typed translation semantics.

The paper discusses properties of the calculus, like type safety, decidability
and stability and state that mechanised proofs for these are currently in
progress.

The problem the authors are tackling is real, and providing ways to facilitate
handling this is worthwhile. I also think the general approach presented by
the authors is interesting and promising. However, in my view, there are
serious problems with the presentation of the paper, to the point where it
is very hard to judge the technical accuracy. And while this is a complex
problem, I cannot shake the feeling that the framework the authors present is
overly complicated. Again, this might be due to the way it is presented.

The authors use a running example to, as they write in the contribution
paragraph, demonstrate the expressive power of the external language as well
detail its semantics. This description points to the core of the problem with
the presentation of this paper: an example program serves one or the other
purpose well. In this case, it might show how powerful the language is, but
complicates the explanation of the semantics and translation process. As a
consequence, it takes the authors until Page 8 to fully discuss this one
example. In the mean time, the reader is left with a huge number of forward
references, which make it unnecessarily difficult to understand what is going
on. A large portion of the code in concrete and abstract syntax is concerned
with the details of describing and dealing with the regular expressions, but
these are orthogonal to the contributions of the paper, but add clutter and
obfuscate the relevant bits. Using a sequence of very simple example programs
to explain the features of the framework would be a lot better. The framework
is complex (maybe a bit more than necessary), so an incremental introduction
is all the more important.

Figure 8 and 9 take up a lot of space, the content is hard to parse and
doesn't actually add all that much information. The important bits could be
presented without listing the full definitions of both tycons.

Review #5B

Modified 29 Aug 2014 5:20:35am EDT

Plain text

Overall merit (?)

D. Reject

Confidence (?)

Z. I am not an expert; my evaluation is that of an informed outsider

Comments for the Authors

I regret to say, but I couldn't make heads or tails of this paper.
The abstract and the introduction provide at best a vague idea of the
goals of the paper (the language is too abstract; certainly some
examples that drive the discussion would help). As an example from the
abstract "We organize each fragment around a type constructor, as is
usual practice when describing type systems, and sidestep the
difficulties of abstract syntax by statically delegating semantic
control over a small, fixed abstract syntax in a type-directed manner
to static logic associated with a relevant type constructor." The
following pages then basically consist of sequences of definitions,
containing lots of forward references. No motivation is given, the
discussion quickly becomes very technical. The typing rules (especially in Figure 11) look daunting. After a while I felt completely lost.

Review #5C

Modified 6 Sep 2014 11:13:41am EDT

Plain text

Overall merit (?)

D. Reject

Confidence (?)

Y. I am knowledgeable in this area, but not an expert

Comments for the Authors

This paper presents a core calculus, called the "actively typed"
lambda calculus (@lambda) for safely composing type system fragments.
The calculus allows extension of the type system via addition of type
constructors (tycons) -- technically the typing judgment is indexed by
a tycon context. The language is organized into an external language
EL that elaborates/translates into an internal language IL. Each type
constructor defines the semantics of its associated operators via
functions written in a static language SL, which also ascribes a kind
to all types from EL.

The authors give two examples of how this calculus can be used to
define fragments for labeled product types and for regular strings.

This goals of this work are important: we do need language frameworks
in which we can grow a language *and* be guaranteed various
metatheoretic properties of not just the new dialect, but also that
two dialects can be composed without violating the metatheory
established for each one. Composability and abstraction are key and
this paper takes a serious stab at the problem.

Unfortunately, the paper presents the metatheory for the calculus in a
"semi-formal" manner. That is, there are statements of a variety of
desired theorems (e.g., type safety, representational consistency,
unicity, decidable type checking, stable typing, etc.) but the authors
haven't completed the proofs for these: "Proving the theorems below
completely rigorously given the normalization semantics we have
presented is difficult" and they are working on mechanized proofs with
different style of specifying semantics. In a calculus this complex,
and with the kinds of properties the authors aim to prove, I don't
think the proofs are going to be at all straightforward and

may
necessitate changes to the calculus. At this point, this paper is
simply not ready for publication.

Review #5D

Modified 6 Sep 2014 11:45am EDT

Plain text

Overall merit (?)

D. Reject

Confidence (?)

Y. I am knowledgeable in this area, but not an expert

Comments for the Authors

The paper proposes a formalism (@lambda) as a technical framework to unify programming language extensions and ensure that, modulo certain constraints, formal properties established for a set of features continue to hold when the language is extended with new features. The suggestion if executed successfully will constitute a landmark for a real theoretical (and practical) advance in our trade.

However, the main weakness of the paper is that it is a blueprint of a master plan for a research that still needs to be completed. The paper recognize this when it comes to the "meat", quoting section 3 titled "Metatheory of @lambda":

"We can only present the theorems and lemmas below semi-formally. Proving the theorems below completely rigorously given the normalization semantics we have presented is difficult, because it often requires us to reason about intermediate terms, rather than sub-terms. [...] We hope to submit mechanized proofs using a more suitable specification style as an artifact, and have started, but not completed, this work using Coq as of submission."

I would argue that given the impact of the promised result, the formal proofs *are* the main contribution, not the intuitions as the paper suggests at the beginning of section 3.

In its current form, the paper is essentially a pile of formal definitions the relevance of which are not at all obvious given the lack of formal proofs that may have used them. It is also apparent that a formal proof would have used a completely different presentation, leaving one to wonder how useful the current presentation is.

Response

The authors' response should address reviewer concerns and correct misunderstandings. Please focus on the main questions and issues of the

reviews and keep it as brief as possible (if the reviewers have asked many subsidiary questions, address those in a clearly delimited section at the end of your response). Please do not use the response to report on new material. Reviewers are instructed to focus on the first 500 words of the response. You may write more, but reviewers will treat explanations that extend beyond 500 words as "optional reading." Please make your most important points early in the response. POPL Author Response End Time: http://www.timeanddate.com/worldclock/fixedtime.html?msg=POPL+2015+Author+Response+Ends&iso=20140910T12&p1=179

Save draft **Submit** Cancel 500 words left