Mary Southern <south163@umn.edu>

Working with Parinati

9 messages

Mary Southern <marys@cs.umn.edu> Fri, Jul 12, 2013 at 4:15 PM
To: Zach Snow <z@zachsnow.com>
Cc: Gopalan Nadathur <gopalan@cs.umn.edu>, david.baelde@lsv.ens-cachan.fr

Hello Zach,
I've begun working with the translation of LF, based mostly from the work in your PPDP 2010 paper. I have been able to take the lemma regarding type checking rigid variables and split it into two, one set just in LF and another which lifts terms from the translation. These can then be used in proving correctness of the improved translation which removes redundant type checks. We've also managed to extend the definition of rigidity to include the cascaded case where a term can appear rigidly in the type of another rigid term.
This work has got me using parinati to look at examples and gain more intuition on the subject.I've noticed that the current translation does not seem to apply the full rigidity condition. Specifically where there are abstractions, ([x:tm] F x). Here's an example I used

```
tm : type.
app : tm -> tm -> tm.
abs : (tm -> tm) -> tm.
copy : tm -> tm -> type.
copyApp : {A:tm}{B:tm}{A':tm}{B':tm} copy A A' -> copy B B' ->
                    copy (app A B) (app A' B').
copyAbs : {F:tm -> tm}{F':tm -> tm} ({x:tm}{y:tm} copy x y ->
                     copy (F x) (F' y)) -> copy (abs ([x:tm] F x)) (abs ([x:tm] F' x)).
```

In the type of copyAbs both F and F' appear rigidly, but their checks are not being removed by the translation. I'd like to work on getting this implemented, so I'd appreciate your advice on getting into the code and making changes. Thanks in advance,
Mary Southern

Zach Snow <z@zachsnow.com> Thu, Jul 25, 2013 at 2:34 PM
To: Mary Southern <marys@cs.umn.edu>

Hey Mary. You are right that parinati doesn't currently implement the HO-patterns aspect of rigidity -- I managed to play with most of the examples I wanted to investigate without it, and probably overlooked a few places where I didn't think it would come into play, but actually did.

The current code is here: https://github.com/zachsnow/parinati/

It's really not that great -- I did a lot of work trying to maintain all forms of the translation for experimental purposes that actually ended up making the code less clear and less DRY. If you are looking to only work on the actual "good" translation I would expect that best way forward would be to take only the parts of parinati that are specific to that translation (the lexer and parser to start, though they are very "loose" in what the accept, but expect it to be correct) and start a bit fresh.

-Zach
[Quoted text hidden]

Gopalan Nadathur <gopalan@cs.umn.edu> Tue, Aug 6, 2013 at 11:49 AM
To: Zach Snow <z@zachsnow.com>
Cc: David Baelde <david.baelde@lsv.ens-cachan.fr>, Mary Southern <marys@cs.umn.edu>

Hi Zach,

Been a while since we communicated, although Mary has been sending you some information about work we are doing related to the translation of LF to HH and developing Parinati further.

Mary and I will be working this month on building in the old rigidity check (actually, really a strictness check, the condition is stronger than rigidity) completely into the translation and on extending this check to include a stronger version of strictness that can be used and for which we have proved correctness. In addition to "fixing" the translation in this sense, we would like to use all this work to provide a utility to accompany the Teyjus suite that will take a Twelf spec and query and respond in a way that resembles what Twelf would do, only it would use the translation and tjsim to carry out the process. We have a way of reconstructing Twelf terms based on what Teyjus provides back as answers for which Mary has written up a proof of its correctness to support this. We also have some thoughts of how to formalize and prove properties when types have variables in them that we hope will find their way into this utility.

I have forgotten now how exactly you had structured the code sharing. If you actually have a git repository and would be willing to let us change what is in there, could you please let us know how to do this? Otherwise, Mary and I will set up an svn or git repo for the code and provide you and David permission to also edit stuff there. The latter may be the best way to go in the end since what we have in mind for the system is somewhat broader than the original project you worked on.

We really want to make progress on this code this month so I hope you will be able to respond soon to the question about preferences. We are already working on the code and we will set up some sharing structure ourselves by the end of the week if we do not hear from you before then.

Regards,
-Gopalan

P.S. David, we plan to keep you in the loop with regard to this work for various reasons, including the fact you were involved with the original proofs and that this stuff is of interest insofar as the M2 -> Abella project is still ongoing. Hope this is okay.

[Quoted text hidden]

Mary Southern <marys@cs.umn.edu> Wed, Aug 28, 2013 at 3:08 PM
To: Gopalan Nadathur <gopalan@cs.umn.edu>
Cc: Zach Snow <z@zachsnow.com>, David Baelde <david.baelde@lsv.ens-cachan.fr>

Hi Zach,
I believe I've got the rigidity check working now and would like to try running it on the examples used in your paper with Gopalan and David Baelde and see how the performance compares. Would you be able to explain how those experiments were set up and the results collected? I'm not even sure where I might find the LF specifications used for the experiments. Thanks for your help,
Mary Southern
[Quoted text hidden]

Zach Snow <z@zachsnow.com> Wed, Aug 28, 2013 at 3:29 PM
To: Mary Southern <marys@cs.umn.edu>
Cc: Gopalan Nadathur <gopalan@cs.umn.edu>, David Baelde <david.baelde@lsv.ens-cachan.fr>

Hey Mary. I have some code hanging around for running timing tests that was just too nasty to put in front of anyone else. They are attached -- apologies.

The basic process is:

1. Write a fully explicit Twelf signature; I cheated a bit and wrote implicit signatures and then asked Twelf to spit them back out in fully explicit form without infix operators.

2. Add a single %solve directive, `main`; for instance:

%solve main : reverse (cons z (cons z (cons z (cons z nil)))) L.

3. Copy and paste that file a half a dozen times, changing the %solve directive to have larger and larger arguments; this was done by hand.

4. Use `twelf-timing` and `twelf-opt` to time each one using Twelf (basically same, but one enabled the indexing optimization). This generates a bunch of *.twelf.timing files, and also writes to the console. The format is ugly; it's just whatever Twelf spits out.

5. Use `lp-timing.py` to time each one using the translation and tjsim, in a really naive way. This just writes to the console a single number I think.

6. Compare :)

I will try to find all of the example files and pass them along as well.

lp-timing.py

twelf-opt

twelf-timing

-Zach
[Quoted text hidden]

Mary Southern <marys@cs.umn.edu> Wed, Nov 13, 2013 at 2:56 PM
To: Zach Snow <z@zachsnow.com>

Hi Zach,
I was able to set up some of the testing benchmarks and run the code to time them with different sized queries. After looking at the timing results, I'm not sure how to interpret them as is. Even with a simple example such as reversing a list I get times for Teyjus which are two to three times longer than the Twelf time which isn't at all what I expected. Let me know if you have any thoughts about how to interpret this and what other information might be useful for you towards thinking about this.
Mary
[Quoted text hidden]

Zach Snow <z@zachsnow.com> Wed, Nov 13, 2013 at 5:57 PM
To: Mary Southern <marys@cs.umn.edu>

Hey Mary. I don't really know off hand, do you want to send me a link to a repo and I will try to check it out?

-Zach
[Quoted text hidden]

Mary Southern <marys@cs.umn.edu> Wed, Nov 13, 2013 at 6:08 PM
To: Zach Snow <z@zachsnow.com>

Zach,
argh, sorry, that was a draft from this morning I meant to delete and not send. I decided to look into things more and try to come up with a more clearly defined question. I'll probably send you a longer more descriptive email about it tonight. I think there might be a problem arising from using CPU time for Twelf and clock time for Teyjus but I want to take time to look at exactly what is happening.
Mary

[Quoted text hidden]

Zach Snow <z@zachsnow.com> Wed, Nov 13, 2013 at 6:09 PM
To: Mary Southern <marys@cs.umn.edu>

(Incidentally, with some versions of reversing a list behave differently than others; in particular the `rev_build` set of examples has Twelf beating Teyjus in my raw data until you hit large enough sizes that Twelf runs into too much gc).

-Zach
[Quoted text hidden]