

Composing Bidirectional Programs Monadically / Ungenerators

11 messages

Harrison Goldstein <hgo@seas.upenn.edu>

Mon, Aug 30, 2021 at 5:36 PM

To: d.a.orchard@kent.ac.uk, Li-Yao Xia <xialiyao@seas.upenn.edu>, meng.wang@bristol.ac.uk, sf16540@my.bristol.ac.uk
Cc: Benjamin Pierce <bcpierce@cis.upenn.edu>

Hi all,

For those of you who did, thank you so much for stopping by my poster at ICFP! I wondered if I could pick your brains a bit more about *Composing Bidirectional Programs Monadically* and how exactly I should be using it in my Ungenerators work. I'm hoping to write a paper soon, so any feedback would be super helpful. (If you need a refresher, [here](#) is my poster.)

I have two main questions that I'd love some input on:

- What are the right properties to prove about ungenerators? At ICFP I mentioned that the main property I settled on was a forward round-tripping property, which is essentially
$$x \text{ `elem` range } g \implies \text{purify } g \ x == \text{Just } x$$
and I originally said that I didn't think that the equivalent backward round tripping property was interesting, but now I'm not sure. My ultimate goal is to be able to prove some properties about generators of type $\text{forall } g. \text{BiGen } g \implies g \text{ a } a$ which give confidence that specific instances will be correct in some way.
- You briefly mention that your definition of Profunctor can be explained as a categorical profunctor whose first mapping is from the Kleisli category of the Maybe monad, but I wondered if you thought there was anything deeper to explore there. The Maybe is obviously necessary to write anything interesting in the backward direction, but does feel a little out of place.

I'm also happy to hear any other feedback you have on the poster, or the idea in general.

Thanks again, it's been really fun working with these ideas!

Best,

Thu, Sep 9, 2021 at
10:12 AM

Samantha Frohlich <s.frohlich.2016@my.bristol.ac.uk>

Reply-To: s.frohlich.2016@my.bristol.ac.uk

To: Harrison Goldstein <hgo@seas.upenn.edu>

Cc: Dominic Orchard <d.a.orchard@kent.ac.uk>, Li-Yao Xia

<xialiyao@seas.upenn.edu>, Meng Wang <meng.wang@bristol.ac.uk>, Benjamin
Pierce <bcpierce@cis.upenn.edu>

Hi Harry,

Sorry for the delay in my response, I've simply been having too much fun playing with your Ungenerators! Thank you very much for sharing your code, it's been a blast. I'm currently working with the Monadic BX authors to add more examples for a journal version of the paper, and it is interesting how different your approach to building upon this work has been.

I've got some general thoughts about Ungenerators, then stuff relating to your first question and round tripping (Dominic is best placed to answer your second question), and finally I thought you might like to hear about what I have been doing! For this, I'll be referencing some code that I have put together and you can find attached (Ungen+BxThoughts.zip). It includes a file where I played with Ungenerators (Ungenerators.hs) in the context of the current state of Monadic BX (BidirectionalProfunctors.hs / PartialProfunctors.hs), which now uses Data.Profunctor, and code that I have been working on for the journal paper (the rest).

So, Ungenerators! I think they are pretty cool, although - to start on a light note - I think you have missed a trick with the name: you should call them Bidirectional UnGenerators or BUGs for short! But seriously, as someone that loves embedding, I really like the way you have taken a tagless final style approach, providing the different directions as different instances/semantics. One thing I would advise for the paper is making that clear and showing examples of running the BUGs with different instances. Maybe it was me being silly, but it took me ages to work out where the ungen bit was. You can see all my exploring in Ungenerators.hs. I'm a commenter so hopefully that can give you insight into how someone would unpack your code.

When it comes to your first question and round tripping I have a few insights for you:

- You will not be able to prove your soundness property compositionally. This is for the same reason that in the paper soundness cannot be proved

compositionally for BiGens. I've gone into lots of detail on this in Ungenerators.hs

- If, however, you constructed its complete counter-part I think that it is compositional. In Ungenerators.hs I've done a few cases of the compositionality proof including new cases for uniform and select. Since the cases I adapted from Monadic BX proofs followed basically the same beats, I didn't bother doing all the proofs as I'm pretty convinced it will work.
- Even though you might not be able to formally prove some round tripping I like your testing based approach as it gives confidence by ruling out some clearly bad things like badGenTree.
- When it came to me being convinced that these properties told me something about Ungenerators the abstraction barrier between me thinking about Ungenerators and these properties specialising to different instances was too much to convince me, so I dabbled with creating you versions of soundness and completeness that specialised to the Seq instance. These follow the same patterns proving-wise as the soundness/completeness for the gen/check instances: soundness cannot be proved compositionally, but I think completeness will be fine. Perhaps them being so similar shows that the soundness/completeness was enough to tell me about the behaviour of Ungenerators, but I think in the paper you'll have to convince some sceptics.

As it might inspire some Ungenerator ideas, I'll show you what I have been doing. Maybe you also have some thoughts about what I have done? I took a completely different approach to building upon the Monadic BX work: where you specialised the Monadic Prof type class; I played about with creating different Monadic Profs and seeing what they can do. I think the best way to explain what I have done is share with you some [slides](https://docs.google.com/presentation/d/1CGSshhpqO4zVCqtJRRbDYCX3_KG0Eyc7jwBLCRmcADE/edit?usp=sharing) (https://docs.google.com/presentation/d/1CGSshhpqO4zVCqtJRRbDYCX3_KG0Eyc7jwBLCRmcADE/edit?usp=sharing). The speaker notes are fairly comprehensive and for the full experience you can click next when a word is in bold. I've also included my code in the attached zip.

Let me know if you have any questions,

Best wishes,

Sam

[Quoted text hidden]

 Ungen+BxThoughts.zip
37K

Mon, Sep 13, 2021 at 9:30 PM

Harrison Goldstein <hgo@seas.upenn.edu>

To: s.frohlich.2016@my.bristol.ac.uk

Cc: Dominic Orchard <d.a.orchard@kent.ac.uk>, Li-Yao Xia <xialiyao@seas.upenn.edu>, Meng Wang <meng.wang@bristol.ac.uk>, Benjamin Pierce <bcpierce@cis.upenn.edu>

Hi Sam,

Thank you so much for the detailed response! I'm just acknowledging this now so you know I've looked at it --- I'll need a few more days to read through your contributions. This all looks like great feedback, and I'm really excited to dig into it.

Best,
Harry

[Quoted text hidden]

Samantha Frohlich <s.frohlich.2016@my.bristol.ac.uk> Tue, Sep 14, 2021 at 8:11 AM

Reply-To: s.frohlich.2016@my.bristol.ac.uk
To: Harrison Goldstein <hgo@seas.upenn.edu>
Cc: Dominic Orchard <d.a.orchard@kent.ac.uk>, Li-Yao Xia <xialiyao@seas.upenn.edu>, Meng Wang <meng.wang@bristol.ac.uk>, Benjamin Pierce <bcpierce@cis.upenn.edu>

Dear Harry,

You are very welcome. I hope my feedback is helpful!

Best,
Sam

[Quoted text hidden]

Harrison Goldstein <hgo@seas.upenn.edu> Thu, Sep 23, 2021 at 8:47 PM

To: s.frohlich.2016@my.bristol.ac.uk

Hi Sam,

I've finished the fellowship application that I was working on, and I finally got around to reading your ungenerators development. It looks fantastic! Some thoughts, questions, etc.

- I can't believe I missed "BUGs"! That's so clever! I'm 100% going to start using that name within my lab (although calling them that in a paper might end up confusing some of the testing folks...)

- The new formulation of profunctors looks really clean. Let me check if my understanding is correct: instead of making your profunctors "natively" partial, you're using the standard definition of Profunctor and then explicitly stating that `comap` maps into the Maybe Kleisli category? Or am I overcomplicating things?
- Regarding the correctness properties and proofs:
 - Your observations about my original correctness property seem correct to me: my property is a "forward" soundness property, which won't be compositional for the same reasons that your bigenerators aren't. I'm not sure how much that bothers me though, since (1) it plays nicely with testing, and (2) it could be proved on a case-by-case basis if needed. Have I missed something important, or does that seem fair to you?
 - After thinking more about it, I agree that the completeness property is also important (I had incorrectly said that it wasn't at ICFP). As you showed, that should be compositional though, so I think that should be relatively straightforward to prove.
 - I had initially resisted using `SeqGen` and `SeqUnGen` to state my correctness properties because I thought that they might be too weak - ideally I'd like to show that if XYZ properties hold for a given `BiUnGen`, then something is true of *any instantiation*. My gut says that if my "generator soundness" is not quasi-compositional but forward round-tripping for parsers is, then it doesn't seem likely that forward round-tripping implies generator soundness. On the other hand, forward round-tripping might be good enough in practice, so I might be looking for properties that are too strong.

I'm going to play around with the correctness properties a bit more and try to get a handle on exactly what I can say about arbitrary instantiations, given certain properties of the uninstantiated `BiUnGen`. I'll keep you posted as that develops.

I'd love to chat at some point to hear a bit more about the new monadic profunctor design and work through a couple of proofs/examples, but I'm sure you're busy so no pressure/rush.

Best,
Harry

[Quoted text hidden]

Samantha Frohlich <s.frohlich.2016@my.bristol.ac.uk>

Mon, Oct 4, 2021 at
3:03 PM

Reply-To: s.frohlich.2016@my.bristol.ac.uk

To: Harrison Goldstein <hgo@seas.upenn.edu>

Thank you for your patience with my response. Teaching just started back with us so it has been a bit chaotic.

I've inlined my responses to your points below, but I'd be more than happy to chat! As you are in the US, I'm guessing the afternoon UK time would suit best? This week, I can do tomorrow (5th Oct), or Wed afternoon. Then I'm away on holiday in Scotland the next week, but more than happy to meet the week after, where again the Monday (18th Oct) and Tuesday (19th Oct) afternoons work best. If none of them work, just let me know what would be better for you and we can try and arrange something :-D

On Thu, Sep 23, 2021 at 9:48 PM Harrison Goldstein <hgo@seas.upenn.edu> wrote:

Hi Sam,

I've finished the fellowship application that I was working on, and I finally got around to reading your ungenerators development. It looks fantastic! Some thoughts, questions, etc.

- I can't believe I missed "BUGs"! That's so clever! I'm 100% going to start using that name within my lab (although calling them that in a paper might end up confusing some of the testing folks...)



- The new formulation of profunctors looks really clean. Let me check if my understanding is correct: instead of making your profunctors "natively" partial, you're using the standard definition of Profunctor and then explicitly stating that `comap` maps into the Maybe Kleisli category? Or am I overcomplicating things?

We now use the standard profunctor class, yes, then we use the `toFailure` class to inject the partiality and allow for the definition of `comap`, which manipulates our first profunctor parameter contravariantly using arrows of the form `(a -> Maybe a)`. If that corresponds to mapping into the Maybe Kleisli category, then yes you have understood right. While I love to dabble in category theory, I prefer to use a more operational model of things in my head because I'm way too forgetful for its loaded terminology. For more insight, you can also check out `PartialProfunctors.hs` in the zip I sent you as it shows its composition, and definition along with a few laws about how it should behave. We can also discuss this in more detail at our video meeting.

- Regarding the correctness properties and proofs:
 - Your observations about my original correctness property seem correct to me: my property is a "forward" soundness property, which

won't be compositional for the same reasons that your bigenerators aren't. I'm not sure how much that bothers me though, since (1) it plays nicely with testing, and (2) it could be proved on a case-by-case basis if needed. Have I missed something important, or does that seem fair to you?

That seems fair to me. The reason the Composing BX Monadically paper goes into such detail about compositional proof is because unlike alternative methods for composing BXs (like lenses), programs made this way are not guaranteed to be correct by construction. Thus there was demand to lighten the proof burden.

- After thinking more about it, I agree that the completeness property is also important (I had incorrectly said that it wasn't at ICFP). As you showed, that should be compositional though, so I think that should be relatively straightforward to prove.

I have also already done the most interesting cases for you ;-P

- I had initially resisted using SeqGen and SeqUnGen to state my correctness properties because I thought that they might be too weak -- ideally I'd like to show that if XYZ properties hold for a given BiUngen, then something is true of *any instantiation*. My gut says that if my "generator soundness" is not quasi-compositional but forward round-tripping for parsers is, then it doesn't seem likely that forward round-tripping implies generator soundness. On the other hand, forward round-tripping might be good enough in practice, so I might be looking for properties that are too strong.

I'm confused by your last couple of sentences. I'd consider generator soundness to be a specific instance of a forwards round tripping property, not something implied by it. I see them as two terms that can describe the same thing, with one just being more general. Also something not being quasi-compositional doesn't mean it cannot be proved. Compositionality / quasi-compositionality are just neat ways you can prove something. You can certainly have soundness / forwards round tripping, and as you say before you can test it or prove it case by case, you just cannot prove it compositionally.

I'm going to play around with the correctness properties a bit more and try to get a handle on exactly what I can say about arbitrary instantiations, given certain properties of the uninstantiated BiUngen. I'll keep you posted as that develops.

Can't wait to see where you take it! Also I wasn't necessarily against specialising your instance to prove something. I just worried that something fishy would be going on, but if you can show that there isn't, then that is a really elegant and lovely way of going about it!

[Quoted text hidden]

Harrison Goldstein <hgo@seas.upenn.edu>

Mon, Oct 4, 2021 at 9:32 PM

To: s.frohlich.2016@my.bristol.ac.uk

Teaching is getting hectic for me too, so I don't think I'll have time to meet this week. Let's do October 19th at 10am EDT / 3pm BST? An hour earlier or later would also be fine.

Enjoy your holiday!

[Quoted text hidden]

Samantha Frohlich

Tue, Oct 5, 2021 at 9:24 AM

<s.frohlich.2016@my.bristol.ac.uk>

Reply-To: s.frohlich.2016@my.bristol.ac.uk

To: Harrison Goldstein <hgo@seas.upenn.edu>

Nice! I've just sent you a zoom invite for then :-D

[Quoted text hidden]

Harrison Goldstein <hgo@seas.upenn.edu>

Tue, Oct 5, 2021 at 4:11 PM

To: s.frohlich.2016@my.bristol.ac.uk

Great! Talk to you then.

[Quoted text hidden]

Samantha Frohlich

Tue, Oct 19, 2021 at 3:00 PM

<s.frohlich.2016@my.bristol.ac.uk>

Reply-To: s.frohlich.2016@my.bristol.ac.uk


To: Harrison Goldstein <hgo@seas.upenn.edu>

It was great to chat to you! Here are my notes from the meeting.

Best,

Sam

[Quoted text hidden]

 **HarryUngenMeeting.txt**
3K

Harrison Goldstein <hgo@seas.upenn.edu>

Tue, Oct 19, 2021 at 4:07 PM

To: s.frohlich.2016@my.bristol.ac.uk

Thanks Sam! It was great chatting with you too. I'll be in touch next month!

Best,
Harry

[Quoted text hidden]