Gmail James Riely <jriely@gmail.com>

An example execution

10 messages

James Riely <jriely@gmail.com> Tue, Apr 3, 2018 at 3:58 PM
To: jeehoon.kang@sf.snu.ac.kr, gil.hur@sf.snu.ac.kr, orilahav@mpi-sws.org, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>
Cc: Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Jeehoon, Chung-Oil, Ori, Viktor and Derek,

Alan and Radha and I have been trying to differentiate various speculative models and we came up the following example. It seems that this execution is allowed by your promising semantics. Can you let us know what you think? Perhaps I've missed some subtlety of the semantics.

Best wishes,
James

```
Thread s:
  a:=x          // a==1
  y:=a
Thread t:
  b:=z          // b==1
  c:=b?y:1      // c==1
  x:=c
Thread u:
  z:=1
```

Thread `t` can also be written

```
  b:=z          // b==1
  if (b==1)
    c:=y        // c==1
    x:=c
  else
    x:=1
```

The transitions of the threads are as follows:

```
s0 --R(x,v)--> s1 --W(y,v)--> s2
t0 --R(z,0)--> t1 --W(x,1)--> t2
t0 --R(z,1)--> t3 --R(y,v)--> t4 --W(x,v)--> t5
u0 --W(z,1)--> u1
```

To get the result, first execute `u` to get message `<z:1@1>`.
Then `t` promises `<x:1@1>`, which it can fulfill by reading `b/z==0`.
Then execute `s` to get message `<y:1@1>`.

Then execute `t`, reading `b/z==1` and `c/y==1` and fulfill the promise by writing `<x:1@1>`.

--
James Riely jriely@gmail.com jriely@cs.depaul.edu http://fpl.cs.depaul.edu/jriely/

Chung-Kil Hur <gil.hur@sf.snu.ac.kr> Tue, Apr 3, 2018 at 6:29 PM
To: James Riely <jriely@gmail.com>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Dear James,

Yes, it is allowed in the way you described.

Best,
Gil
[Quoted text hidden]

James Riely <jriely@gmail.com> Wed, Apr 4, 2018 at 9:34 AM
To: Chung-Kil Hur <gil.hur@sf.snu.ac.kr>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Thanks!
[Quoted text hidden]

Chung-Kil Hur <gil.hur@sf.snu.ac.kr> Wed, Apr 4, 2018 at 9:55 AM
To: James Riely <jriely@gmail.com>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Dear James,

In your original email, I missed some part.

```
s0 --R(x,v)--> s1 --W(y,v)--> s2
t0 --R(z,0)--> t1 --W(x,1)--> t2
t0 --R(z,1)--> t3 --R(y,v)--> t4 --W(x,v)--> t5
u0 --W(z,1)--> u1
```

Here, you used "v" in several places. I didn't notice it before.
In the promising semantics, "v" can be only 0 or 1 because that's what the thread s can read from X.
Please let me know if you intended something else.

Thanks.

Best,
Gil
[Quoted text hidden]

James Riely <jriely@gmail.com> Wed, Apr 4, 2018 at 10:08 AM
To: Chung-Kil Hur <gil.hur@sf.snu.ac.kr>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Hi Gil,
Yes, I was just using v as ranging over {0,1}.
It seems that the execution is allowed, regardless.
Is that right?
James

[Quoted text hidden]

Chung-Kil Hur <gil.hur@sf.snu.ac.kr> Wed, Apr 4, 2018 at 10:10 AM
To: James Riely <jriely@gmail.com>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Yes, you are right.
The execution is allowed for both 0 and 1.

Best,
Gil
[Quoted text hidden]

James Riely <jriely@gmail.com> Wed, Apr 11, 2018 at 7:15 AM
To: Brijesh Dongol <brijesh.dongol@brunel.ac.uk>

[Quoted text hidden]

James Riely <jriely@gmail.com> Sat, Dec 8, 2018 at 11:56 AM
To: Viktor Vafeiadis <viktor@mpi-sws.org>, sohachak@mpi-sws.org

Soham and Viktor,
Can you also answer a question about the execution below?
I believe this outcome is allowed by your POPL20 semantics. Is that correct?
Thanks,
James
[Quoted text hidden]

James Riely <jriely@gmail.com> Tue, Mar 17, 2020 at 7:00 AM
To: Chung-Kil Hur <gil.hur@sf.snu.ac.kr>
Cc: Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Hi Gil & friends,
I just saw that promising 2.0 is out. I've not had time to look at it
yet, but I am curious to know up front whether the status of this
example has changed between 1.0 and 2.0.

Please let me know.
Hope you are all well in the crazy time!
Best wishes,
James
[Quoted text hidden]

Ori Lahav <orilahav@gmail.com> Tue, Mar 17, 2020 at 8:04 AM
To: James Riely <jriely@gmail.com>
Cc: Chung-Kil Hur <gil.hur@sf.snu.ac.kr>, Jeehoon Kang <jeehoon.kang@sf.snu.ac.kr>, Ori Lahav <orilahav@mpi-sws.org>, Viktor Vafeiadis <viktor@mpi-sws.org>, Derek Dreyer <dreyer@mpi-sws.org>, Radha Jagadeesan <RJagadeesan@cs.depaul.edu>, Alan Jeffrey <asaj@asaj.org>

Hi James,

There is no difference for this example :)
The differences only concern programs with RMWs.
Also: PS2.0 is weaker than PS1.0 (whatever was allowed is still allowed).

Best regards (and health!),
Ori
[Quoted text hidden]

Gmail James Riely <jriely@gmail.com>

program behavior and visibility in POPL19 paper

3 messages

Soham Chakraborty <sohachak@mpi-sws.org> Mon, Dec 10, 2018 at 8:40 AM
To: James Riely <jriely@gmail.com>
Cc: Viktor Vafeiadis <viktor@mpi-sws.org>

Dear Prof. Riely,

I have combined your mails along with the answers. Please let us know if you have further queries.

Best Regards,

Soham

==========

Mail 1

Hi Soham and Viktor,
I am just reading your POPL20 paper and trying to understand the notion of "visible".
On page 5, you say "e is invisible whenever there is a conflicting write event in its (po ∪ jf)+-prefix that does not have an equal write in the *same execution branch as e*". It seems by this definition that Ld(X,1) in figure 1a is invisible, since St(X,1) is in its (po ∪ jf)+-prefix, but there is no St(X,1) that is po-related to Ld(X,1)

Answer: St(X,1) is in the other thread and not in conflict with Ld(X,1). Note that Ld(X,1) has St(Y,1) in its (po U jf)^+-prefix which is in conflict and Ld(X,1) has a po-after St(Y,1) event. These two St(Y,1) events can be equal-writes (ew) and then Ld(X,1) is visible (as shown in Figure 4b).

On page 12, you say "e is visible in an event structure G if all the writes recursively used to justify e that are in conflict with e have some equal write that does not conflict with e". This makes more sense to me, but then the formal definition uses "po=" which is "(po cup po-1)?"

Answer: Consider the example in the pic-1 (attached).
In the event structure a, b, c are visible events as
(w,a) \in G.ew;G.po^{-1} and (w,b) \in G.ew and (w,c) \in G.ew;G.po, that is,
(w,a),(w,b),(w,c) \in G.po^{=}.

I am obviously missing something. Can you help me out?
Thanks,
James

Mail 2

Soham and Viktor,
Can you also answer a question about the execution below?
I believe this outcome is allowed by your POPL20 semantics. Is that correct?

Answer: Yes the following outcome is allowed in our semantics. The event structure and the extracted execution are in pic-2 (attached).

Thread s:
 a:=x // a==1
 y:=a
Thread t:
 b:=z // b==1
 c:=b?y:1 // c==1
 x:=c
Thread u:
 z:=1

Thanks,
James

mail 3

One more thing:

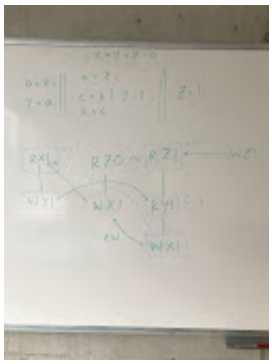
You also say "As a side note, we remark that the surpringly weak outcomes of tests 5 and 10 (forbidden by our models) can also be explained by thread sequentialization, yet Java forbids them." Can you please elaborate? I don't see how inlining can get this program to write x=1 when reading z=0.

Answer: Please check http://plv.mpi-sws.org/weakest/appendix.pdf B1 (page 49-50)

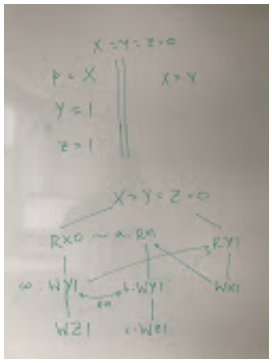
Thanks,
James

---

2 attachments



pic-1.jpeg
1904K



pic-2.jpeg
2127K

James Riely <jriely@gmail.com> Mon, Dec 10, 2018 at 10:15 AM
To: sohachak@mpi-sws.org
Cc: Viktor Vafeiadis <viktor@mpi-sws.org>

Great thanks for the replies.
About Mail 1. Amateur mistake on my part! I was confusing the two notions of conflict: event structure "conflict" and read-write "conflict". St(X,1) in in r-w conflict with Ld(X,1), but not #-conflict.
[Quoted text hidden]
--
James Riely jriely@gmail.com jriely@cs.depaul.edu http://fpl.cs.depaul.edu/jriely/

James Riely <jriely@gmail.com> Thu, Jan 3, 2019 at 12:33 PM
To: Radha Jagadeesan <RJagadeesan@cs.depaul.edu>

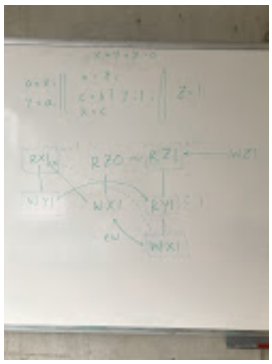
The comments on TC5 and TC10 will amuse you. These are forbidden thin air behaviors from the JMM that are allowed by certain program transformations.

See p50 (!) of appendix that they link to below.

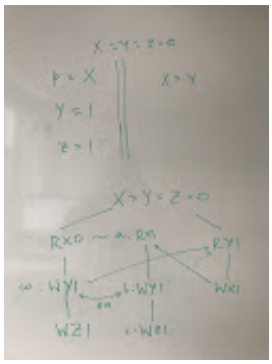
These happen by performing transformations that:
1) introduce conditionals
2) inline two threads on both sides of the introduced conditional
3) choose different orders for the two threads for the two sides of the conditional.

Quite a bizarre sequence of transformations!
[Quoted text hidden]
[Quoted text hidden]

2 attachments



pic-1.jpeg
1904K



pic-2.jpeg
2127K