GSOL: A Confluence Checker for Haskell Rewrite Rules

Date Yao Faustin Makoto Hamana

22nd August, 2021

Gunma University, Japan

This Work

- GSOL: a tool to check the confluence of GHC rewrite rules
- The tool uses the Second-Order Laboratory (SOL) [1, 2] for checking the confluence
- GHC has the feature of user-defined **rewrite rules** [3]
- SOL is powerful in verifying the confluence of higher-order rewrite systems

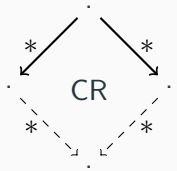
Motivation

```
-- F.hs
module F where

{-# RULES
   "f/0" forall x. f x = 0 ;
   "f/1" forall x. f x = 1 ;
#-}

e = f 99

f :: Integer -> Integer
```

- `"f/0"` and `"f/1"` both match `f 99`.
- If GHC applies `"f/0"`, `f 99` rewrites to 0.
- If GHC applies `"f/1"`, `f 99` rewrites to 1.
- The rules `"f/0"` and `"f/1"` are *non-confluent*.
- GHC does not notice this non-confluence.

GSOL uses SOL to automate confluence checking of user-defined Haskell rewrite rules.

The Notion of Confluence



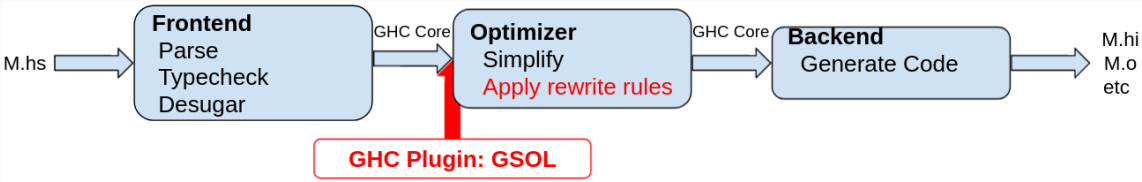
Confluence guarantees uniqueness of normal forms

- **Confluence (CR)** is when any two divergent computation paths are joinable.
- **Local Confluence (LCR)** is when all one step divergences are joinable.
- **Termination (SN)** is when there are no infinite rewrite steps.

GSOL uses **Newman's Lemma** to conclude CR by checking LCR and SN.

Implementation

- GSOL is implemented as a GHC Core plugin.



- GSOL's tasks
  1. Collect the Core rules
  2. Convert the Core rules to SOL rules
  3. Call SOL for checking:
     - SOL performs its checking functions,
     - SOL prints the results to standard output.

Second-Order Laboratory (SOL)

- SOL implements a formal framework of *second-order computation systems*.
- Second-order computation systems use the language of meta-terms:

$$t ::= x \quad | \quad x.t \quad | \quad f(t_1, \ldots, t_n) \quad | \quad M[t_1, \ldots, t_n].$$

- These forms are respectively variables, abstractions, and function terms, and meta-applications.
- The meta-terms have *second-order types*.
- Computation rules are pairs of meta-terms.

Usage

Stand-alone command

```
gsol [PROPERTY] FILENAME
```

- PROPERTY
  - cri local confluence checking
  - sn termination checking
- FILENAME path to the Haskell program file

Web Interface

http://solweb.mydns.jp/

Example: Arrow

- Control.Arrow is a library in GHC
- It offers a way to programming with various computational effects
- It is implemented with the type class:

```
class Category a => Arrow a where
 arr    :: (b -> c) -> a b c
 first :: a b c -> a (b,d) (c,d)
 ...
```

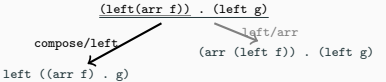
Rewrite rules in Control.Arrow

```
{-# RULES
"compose/arr"     forall f g. (arr f) . (arr g)       = arr (f . g)
"first/arr"       forall f   . first (arr f)          = arr (first f)
"second/arr"      forall f   . second (arr f)         = arr (second f)
"compose/first"   forall f g. (first f) . (first g)   = first (f . g)
"product/arr"     forall f g. arr f *** arr g         = arr (f *** g)
...
 #-}
```

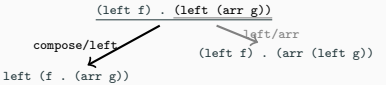
- These rewrite rules are described in Control.Arrow
- NB: The laws assume `a = (->)`

8 critical pairs in the Arrow Laws

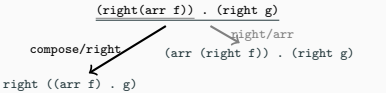
1.



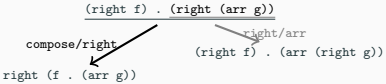
2.



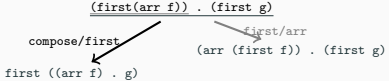
3.



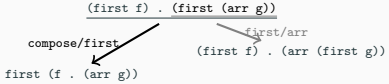
4.



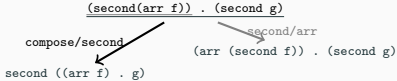
5.



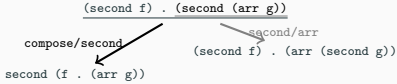
6.



7.



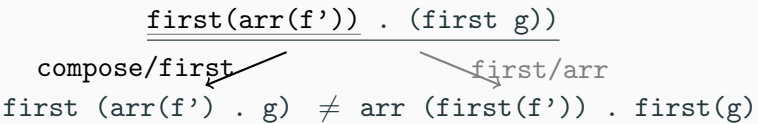
8.



A closer look at a non-joinable critical pair

```
"compose/first"  forall f g. (first f) . (first g) = first (f . g)
"first/arr"      forall f'. first (arr f')          = arr (first f')
```

- Since `(first f)` is unifiable with `first (arr f')` by $\texttt{f} \mapsto \texttt{arr f'}$
  it generates a critical pair:



- Since these are normal forms,
  it shows non-confluence of the rules in Control.Arrow.
- Useful information to redesign the rules.

Conclusion

Future work

- Better handling of types in the translation from Core rules to SOL rules
  - GSOL ignore type constraints
  - Enough for checking critical pairs, but termination checking is weaker

Summary

- GSOL is a tool to check the confluence of Haskell rewrite rules
- GSOL uses the Second-Order Laboratory (SOL) for confluence checking
- Web interface http://solweb.mydns.jp/
- Draft paper http://solweb.mydns.jp/gsol-draft.pdf [4].

References

[1] M. Hamana, "How to prove decidability of equational theories with second-order computation analyser SOL," *Journal of Functional Programming*, vol. 29, no. e20, 2019.

[2] M. Hamana, T. Abe, and K. Kikuchi, "Polymorphic computation systems: Theory and practice of confluence with call-by-value," *Science of Computer Programming*, vol. 187, no. 102322, 2020.

[3] S. P. Jones, A. Tolmach, and T. Hoare, "Playing by the rules: rewriting as a practical optimisation technique in GHC," in *Haskell Workshop 2001*, 2001.

[4] Y. F. Date and M. Hamana, "GSOL: A confluence checker for Haskell rewrite rules." `http://solweb.mydns.jp/gsol-draft.pdf`.