ARI adoption in termCOMP

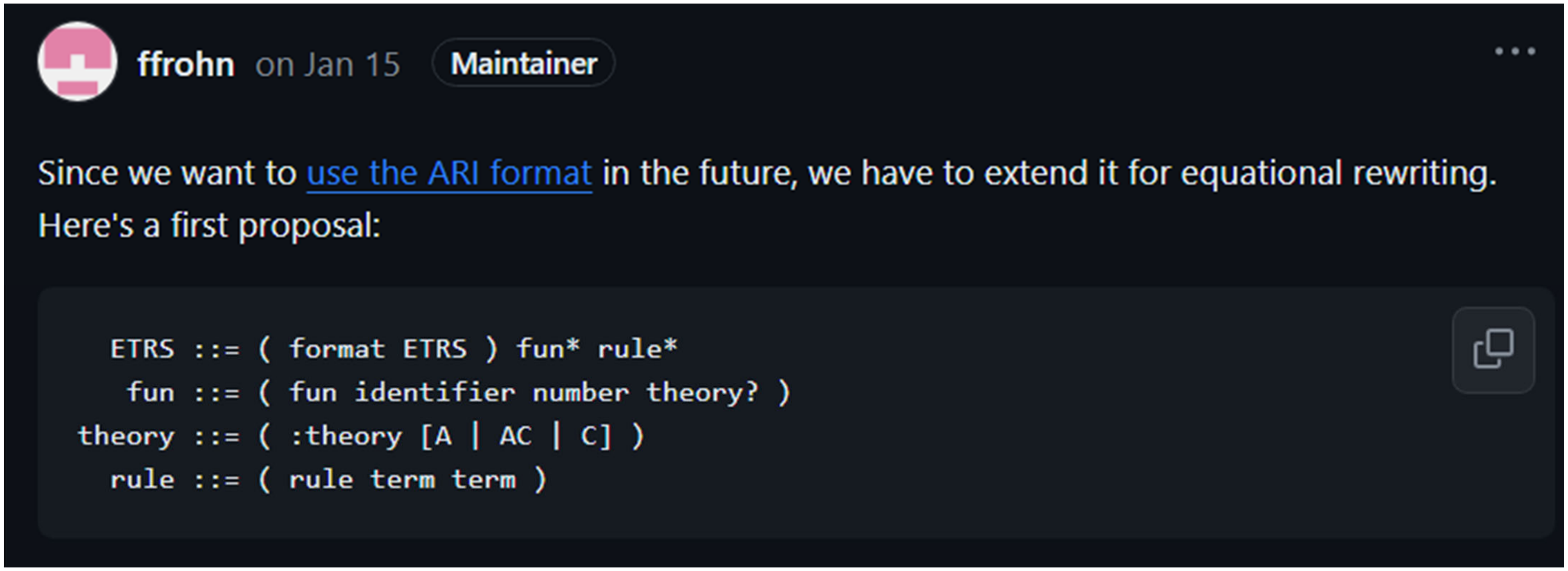
Akihisa Yamada

ARI final meeting 2024/02/22

Overview

- termCOMP has almost decided to adopt ARI for
  - TRS Standard / Innermost / Outermost / CTRS / CSTRS
  - Complexity
- discussion ongoing at GitHub:
  - TRS Equational
  - TRS Relative / Relative Complexity
  - TRS Probabilistic
  - SRS
  - ITS
  - ITRS
  - Higher-order

TRS equational

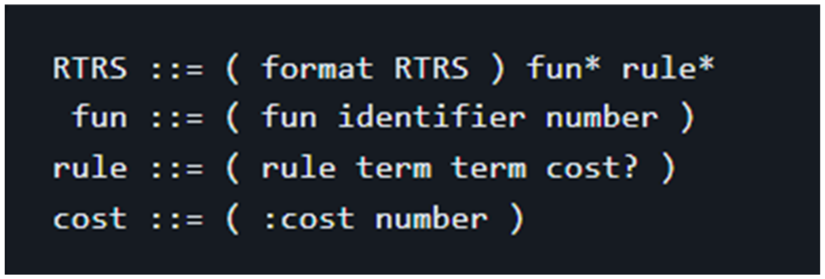


Since we want to use the ARI format in the future, we have to extend it for equational rewriting. Here's a first proposal:

```
  ETRS ::= ( format ETRS ) fun* rule*
   fun ::= ( fun identifier number theory? )
theory ::= ( :theory [A | AC | C] )
  rule ::= ( rule term term )
```

TRS Relative

- option 1: use ":number" and analyze $\mathrm{SN}(\rightarrow_1/\rightarrow_2)$
  - 😄 can reuse ARI infra
- option 2:



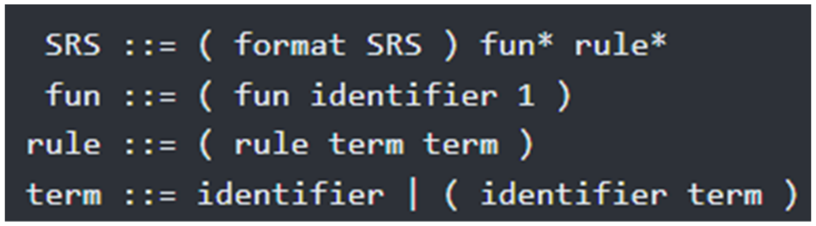
```
RTRS ::= ( format RTRS ) fun* rule*
 fun ::= ( fun identifier number )
rule ::= ( rule term term cost? )
cost ::= ( :cost number )
```

  - 😄 make clear sense in relative complexity
  - 😄 extensible for non-constant cost annotations

SRS

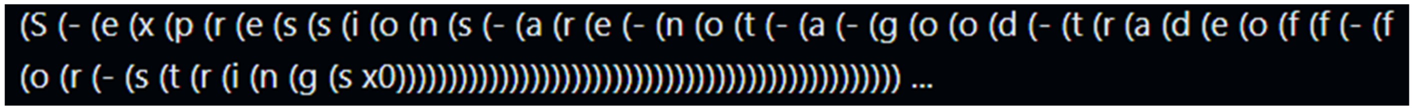
https://github.com/orgs/TermCOMP/discussions/87

ffrohn



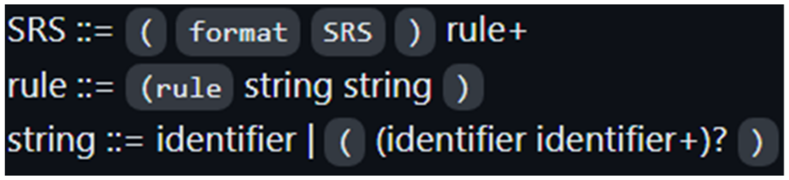
```
  SRS ::= ( format SRS ) fun* rule*
  fun ::= ( fun identifier 1 )
rule ::= ( rule term term )
term ::= identifier | ( identifier term )
```

jwaldmann



```
(S (- (e (x (p (r (e (s (s (i (o (n (s (- (a (r (e (- (n (o (t (- (a (- (g (o (o (d (- (t (r (a (d (e (o (f (f (- (f
(o (r (- (s (t (r (i (n (g (s x0))))))))))))))))))))))))))))))))))))))))))))))))) ...
```

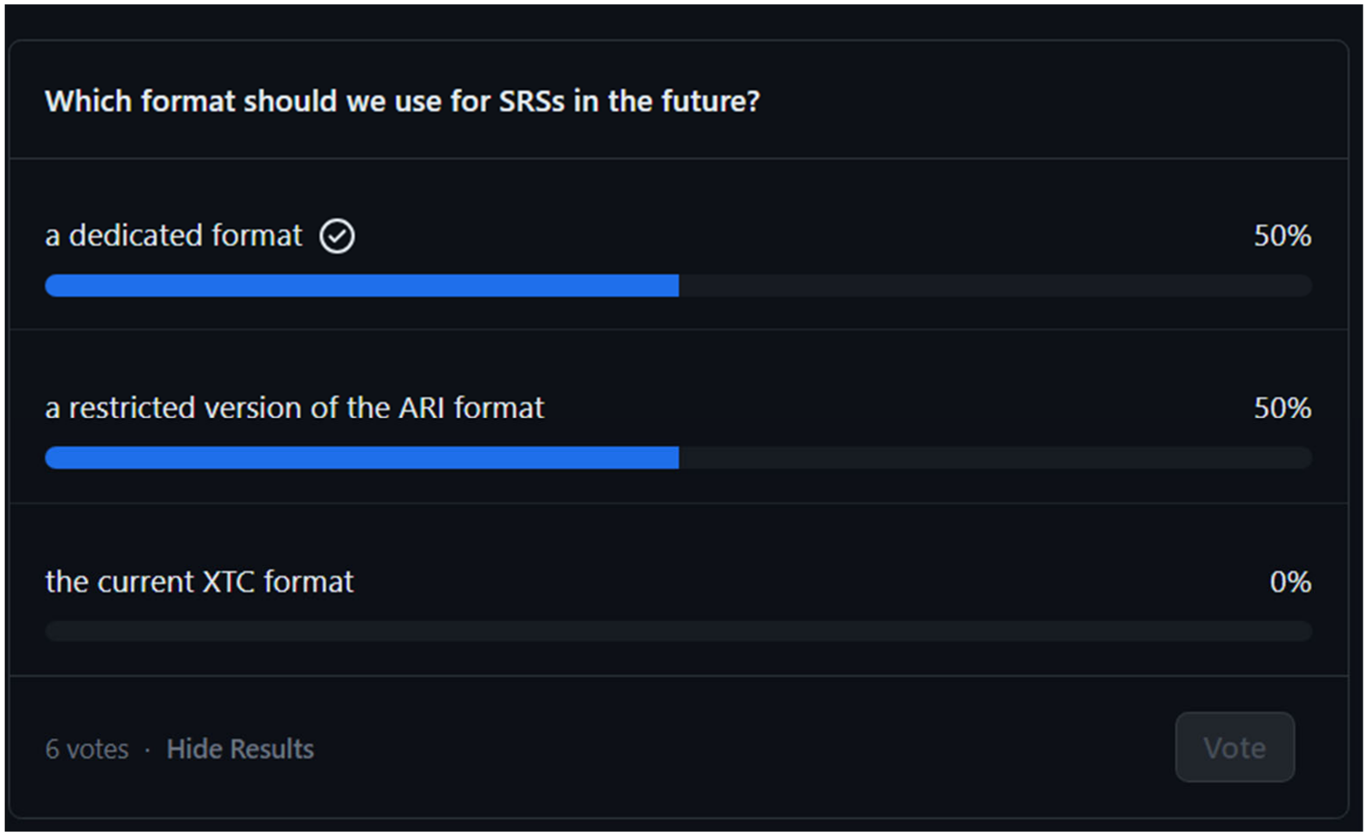
AkihisaYamada



```
SRS ::= ( format SRS ) rule+
rule ::= (rule string string )
string ::= identifier | ( (identifier identifier+)? )
```

My principle: format is syntax. Semantics is up to competition category.

SRS



Which format should we use for SRSs in the future?

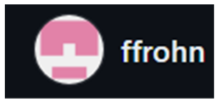
a dedicated format ✓ 50%

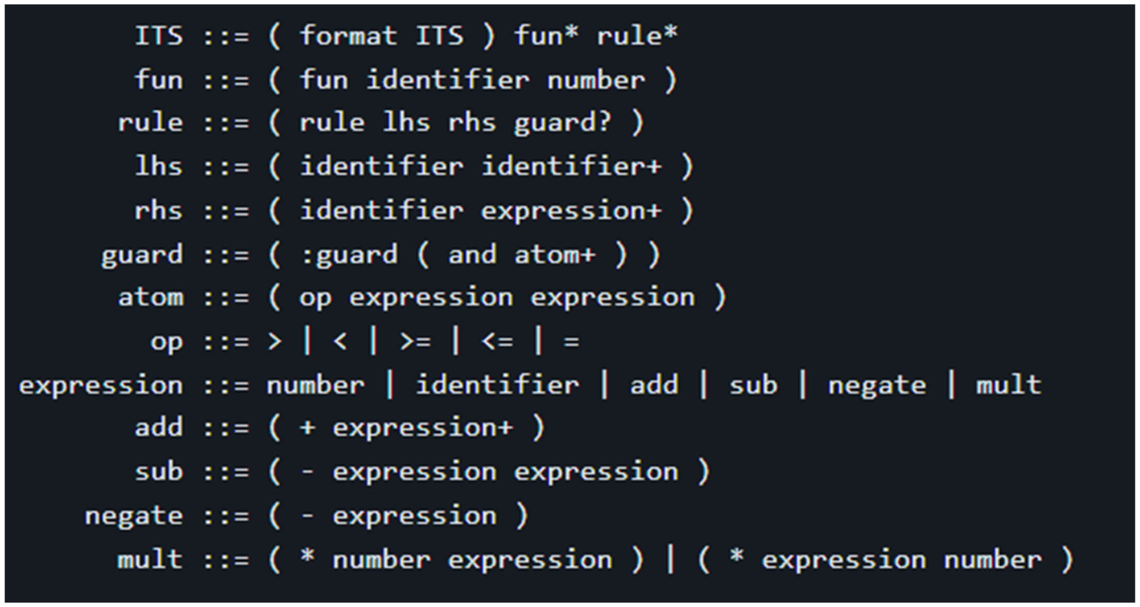
a restricted version of the ARI format 50%

the current XTC format 0%

6 votes · Hide Results

ITS


ffrohn



```
       ITS ::= ( format ITS ) fun* rule*
       fun ::= ( fun identifier number )
      rule ::= ( rule lhs rhs guard? )
       lhs ::= ( identifier identifier+ )
       rhs ::= ( identifier expression+ )
     guard ::= ( :guard ( and atom+ ) )
      atom ::= ( op expression expression )
        op ::= > | < | >= | <= | =
expression ::= number | identifier | add | sub | negate | mult
       add ::= ( + expression+ )
       sub ::= ( - expression expression )
    negate ::= ( - expression )
      mult ::= ( * number expression ) | ( * expression number )
```

😄 Turning (format ITS) to (format LCTRS) (theory Ints) yields a correct LCTRS

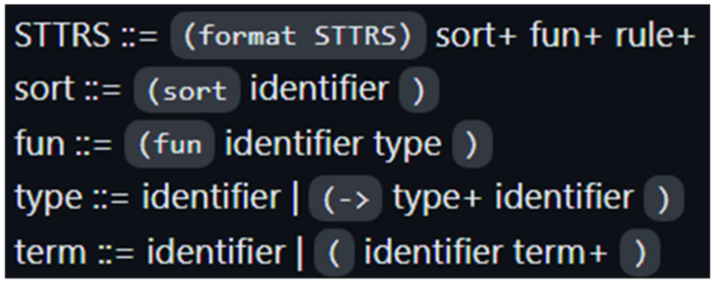
ITRS

😞 termCOMP need to define a restriction of LCTRS

- definitely exclude nasty SMT-LIB features
    - the "_" things
    - let, forall, exists, ite
- probably also Boolean variables

higher order

- Applicative Simply Typed TRS (STTRS) is clear and has potential participants. Why not to have the category?

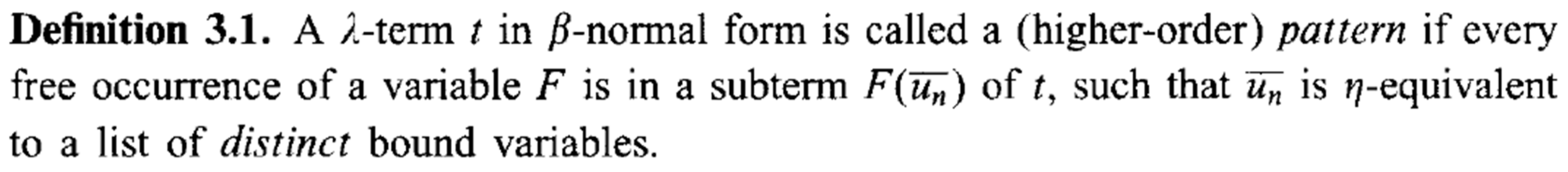


```
STTRS ::= (format STTRS) sort+ fun+ rule+
sort ::= (sort identifier )
fun ::= (fun identifier type )
type ::= identifier | (-> type+ identifier )
term ::= identifier | ( identifier term+ )
```

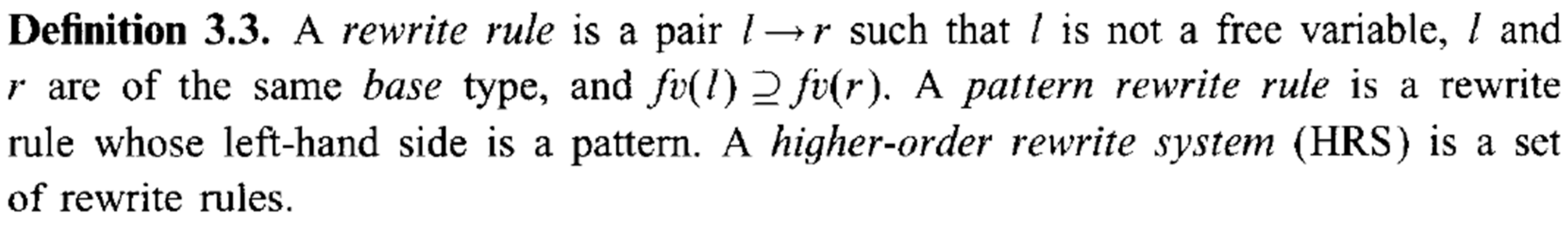
- Can higher-order with λ be rescued?
  - I see no chance in SOL re-joining if the semantics is not "HRS"
  - Wanda can deal with 2nd-order HRS. So I proposed 2nd-order HRS category
  - But Cynthia hates HRS
  - So I don't think there will be any competition on HO with λ in near future.

What is HRS??

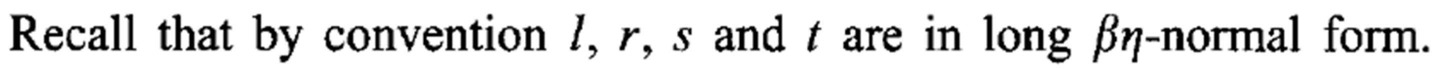
[Mayr & Nipkow, TCS 192 (1998) 3-29] says



Definition 3.1. A λ-term t in β-normal form is called a (higher-order) *pattern* if every free occurrence of a variable F is in a subterm $F(\overline{u_n})$ of t, such that $\overline{u_n}$ is η-equivalent to a list of *distinct* bound variables.



Definition 3.3. A *rewrite rule* is a pair $l \rightarrow r$ such that l is not a free variable, l and r are of the same *base* type, and $fv(l) \supseteq fv(r)$. A *pattern rewrite rule* is a rewrite rule whose left-hand side is a pattern. A *higher-order rewrite system* (HRS) is a set of rewrite rules.



Recall that by convention l, r, s and t are in long $\beta\eta$-normal form.

... and everyone says that rule must be η-long!

Then having STTRS makes duplicates

- All functional programmers will like

```
    (rule (map f (cons x xs))
          (cons (f x) (map f xs)))
```

- but the experts demand

```
    (rule (map (lambda ((x Nat)) (f x)) (cons x xs))
         (cons (f x) (map f xs)))
```

Consequently, proposing STTRS leads to introducing duplicates!
so I even withdraw STTRS

Conclusion

- TRS: Aachen, Tokyo 😄
- ETRS: Aachen, Tokyo 😄
- RTRS: Aachen, Tokyo 😄
- SRS: Aachen, Leipzig, Tokyo 😄
- ITS: Aachen 😄
- ITRS: Aachen, London, Tokyo 😄
- HO: Gunma, Nijmegen, Saclay, Tokyo 🙁

- Q: What will the transformer's license be?