Anubis - Feature #53
[Polysemy]Interpretation goes undetected

08/20/2008 05:05 PM - SpiceGuid -

Status:	Closed	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:	Alain Prouté	**% Done:**	0%
Category:	Design	**Estimated time:**	0.00 hour
Target version:			
Platform:		**Triage Stage:**	
Resolution:	fixed		

Description

The final line:

Has two possible interpretations and yet is compiled without error or warning.

Seemingly, one possible interpretation goes undetected.

Admittedly my code is weird, certainly i have defied some implicit recommandation that the * operator must be associative, that is of the type ($A,$A) -> $A.

<pre>
```
define ($A,$A,$B,$B) -> ($A,$B)
  ($A,$A) -> $A f * ($B,$B) -> $B g
  =
  ($A a1,$A a2,$B b1,$B b2)
  |->
  (f(a1,a2),g(b1,b2)).

define ($A,$A,$B,$B,$C,$C) -> ($A,$B,$C)
  ($A,$A) -> $A f * ($B,$B,$C,$C) -> ($B,$C) g
  =
  ($A a1,$A a2,$B b1,$B b2,$C c1,$C c2)
  |->
  with a  = f(a1,a2),
  with bc = g(b1,b2,c1,c2),
  if bc is (b,c) then
  (a,b,c).

define ($A,$A,$B,$B,$C,$C) -> ($A,$B,$C)
  ($A,$A,$B,$B) -> ($A,$B) f * ($C,$C) -> $C g
  =
  ($A a1,$A a2,$B b1,$B b2,$C c1,$C c2)
  |->
  with ab = f(a1,a2,b1,b2),
  with c  = g(c1,c2),
  if ab is (a,b) then
  (a,b,c).

type Interval($T) :
  empty,
```

```
  between($T lower,$T upper).

define (Interval($T),Interval($T)) -> Interval($T)
  intersect(($T,$T) -> Bool less)
  =
  (Interval($T) a,Interval($T) b)
  |->
  if a is between(a1,a2) then
    if b is between(b1,b2) then
      if less(a2,b1) then empty
      else if less(b2,a1) then empty
      else if less(a1,b1) then between(b1,a2)
      else between(a1,b2)
    else empty
  else empty.

define
  (
  Interval($A),Interval($A),
  Interval($B),Interval($B),
  Interval($C),Interval($C)
  ) ->
  (Interval($A),Interval($B),Interval($C))
  intersect_triple
  (
  ($A,$A) -> Bool a_less,
  ($B,$B) -> Bool b_less,
  ($C,$C) -> Bool c_less
  )
  =
  intersect(a_less) * intersect(b_less) * intersect(c_less).
</pre>
```

History

#1 - 08/21/2008 06:35 PM - Alain Prouté

- Status changed from New to Closed

- Resolution set to fixed

Thank you very much Damien for this remark.

However, your text is not ambiguous, but this is for a purely syntactical reason. Indeed, the binary operation * associates to the right, so that

intersect(a_less) * intersect(b_less) * intersect(c_less)

is read as:

intersect(a_less) * (intersect(b_less) * intersect(c_less))

making the third paragraph:

```
define ($A,$A,$B,$B,$C,$C) -> ($A,$B,$C)
  ($A,$A,$B,$B) -> ($A,$B) f * ($C,$C) -> $C g  = ...
```

useless, as you can check by inhibiting it. If you put a pair of parentheses like this:

```
    (intersect(a_less) * intersect(b_less)) * intersect(c_less)
```

the second paragraph becomes useless instead of the third one.

A similar situation is the term:

```
    1 + 2 * 3
```

which would have two interpretations (1 + 2) * 3 and 1 + (2 * 3) if no precedence/association rule was in effect. (In this case, this is a precedence rule, and an association rule in your example).

Behind this fact, there is undoubtly a problem, which is that precedence/association rules are rules which are in some sens implicit. If the grammar was written without theses rules, we would have to write either (1 + 2) * 3 or 1 + (2 * 3) and 1 + 2 * 3 would become a syntax error.

Recently, I wondered if it was a good idea to remove all precedence/association rules. I will not do it for Anubis 1 anyway, because it would generate big problems in the very big already exiting programs. But for Saunders, I will reduce these rules to a bare minimum. Probably, I will keep 'obvious' rules, like the precedence of * over +, but in any case, the number of rules will be small so that it is easily remembered by users.

A long time ago, I had a bug in a C program, which took me 3 days to discover. I had written something like this (I don't remember exactly):

```
a & b >> c
```

According to the rules of ANSI C this is interpreted as: a & (b >> c), but I was thinking to (a & b) >> c. I'm almost sure that nobody knows the big table of precedences/associations for C by heart.

This is a danger, and I want to reduce this danger in the future at least for Saunders.