CoCoA-5 - Design #997

Using protected variable names for "bound variables" (e.g. for, try...endtry)

18 Jan 2017 11:38 - John Abbott

Status:	Closed	**Start date:**	18 Jan 2017
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Parser/Interpreter	**Estimated time:**	0.99 hour
Target version:	CoCoA-5.3.0	**Spent time:**	0.95 hour

Description

I was surprised the following did not produce any error about protected variables:

```
for QQ := 1 to 5 do println QQ endfor;
try 1+2 uponerror QQ do println QQ endtry;
try 1/0 uponerror QQ do println QQ endtry;
```

Is it worth trying to fix this?

Related issues:

Related to CoCoA-5 - Feature #1003: New syntax for creating poly rings?	**In Progress**	**27 Jan 2017**

History

#1 - 18 Jan 2017 11:44 - John Abbott

The following example does nothing embarrassing:

```
define fn(a) TopLevel QQ; return RingElem(QQ,a); enddefine;
for QQ := 1 to 5 do println fn(QQ); endfor;
```

The point is that **TopLevel** does retrieve the correct variable QQ (namely the protected one containing the field of rationals).

#2 - 18 Jan 2017 13:03 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I note also that "protected names" are allowed as formal parameters for user defined fns: *e.g.* the following works fine

```
define add1(QQ) return QQ+1; enddefine;
```

The important point is that a name may refer to different variables in different contexts, and it is the variable which is protected (rather than the name).

Top-level names are just variables assigned to at top level (and some system protected ones), and names "assigned to" by the **define** command.

Perhaps the best action would be just to make sure that this is clear (somewhere) in the documentation.

#3 - 27 Jan 2017 08:04 - Anna Maria Bigatti

This, not surprisingly, doesn't work ;-)

```
Define FunnyFunc(GBasis)
  return GBasis(ideal(GBasis));
EndDefine;
```

Even though surprinsing this is not seriously ambiguous or dangerous.

#4 - 27 Jan 2017 15:44 - John Abbott

The following works:

```
P ::= QQ[QQ];
-- use PP; --> gives error (phew!) because QQ is protected
qq := indet(P,1);
qq; --> prints "QQ" obviously!
```

Boggle!

#5 - 27 Jan 2017 15:59 - John Abbott

Think about the following valid CoCoA-5 code:

```
P := NewPolyRing(QQ, "define, func, enddefine");
use P;  --> yes, this works!  8-O
```

This creates toplevel variables whose names are the same as keywords; there is no way to access the values of these variables!

#6 - 27 Jan 2017 15:59 - John Abbott

- Related to Feature #1003: New syntax for creating poly rings? added

#7 - 02 Mar 2020 21:55 - John Abbott

- Assignee set to John Abbott

- Target version changed from CoCoA-5.?.? to CoCoA-5.3.0

- % Done changed from 10 to 50

I had forgotten about this "entertaining" issue.

I do not really see anything "dangerous" here; the only slightly surprising fact is that the variable of a top-level **for** loop is **not** itself a top-level variable.

I am inclined to close this issue. Agreed?

#8 - 04 Mar 2020 19:09 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 50 to 100

- Estimated time set to 0.99 h

Anna agrees with the proposal to close this issue.

The problems highlighted here are easy to work around (and you have to be a vandal to encounter them anyway ;-)