CoCoA-5 - Feature #131

Conversion from bool to INT

16 Apr 2012 16:36 - John Abbott

Status:	Closed	**Start date:**	16 Apr 2012
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	CoCoA-5 function: new	**Estimated time:**	0.00 hour
Target version:	CoCoA-5.0.3	**Spent time:**	2.45 hours

Description

Alessio would like a conversion from {false,true} to {0,1}.
Wikipedia mentions a function called "Iverson bracket" which does just that, but its syntax is not suitable for us.

Do we wabt such a function? What should it be called?
Should it be automatic?

History

#1 - 18 Apr 2012 15:46 - John Abbott

Re Automatic conversion:

- Advantages: no need to choose a name, compactness
- Disadvantages: possible ambiguity, unsafe for incautious users, perhaps hard to comprehend

Clearly 1+N>0 is quite different from 1+(N>0).

Would we allow an expression like (N>0)+(M>0) ?

The meaning of (N>0)>(M>0) is clear.
The meaning of (N>0)>M is fairly clear.
What about the expression N>0>M -- maybe it should be forbidden, or it should produce a warning? After all what would you expect 0<i<N to mean?

#2 - 18 Apr 2012 15:58 - John Abbott

We cannot use the Iverson bracket syntax directly because it already has a meaning: [N>0] will produce a list containing a single boolean value.

We could use something similar though. I am thinking of ?[...] or perhaps [?...]. We would presumbaly have to define ?[as a lexeme, unless we wanted to have ? as a unary operator.

The use of ? appeals to me; moreover it is an available character (except at the very start of a command).

I also think that using a "bracket syntax" should avoid complications similar to those I raised in my previous posting.

#3 - 21 May 2012 11:48 - John Abbott

Though I didn't say so explicitly previously, I believe that a truly **automatic** conversion from BOOL to INT is likely to cause more trouble than it's

worth.

I recall that C and C++ have their own approach, via the ?: operator pair. The advantage of the C style syntax is that one can express very easily "if *cond* is true then give value *A* otherwise give value *B*". Just for clarity the C++ code would be cond?A:B. Note that we cannot employ the same identical syntax because : already has a definition in CoCoA language.

Using the syntax proposed in my previous post one would have to write B + (A-B)*[?cond] which is certainly not as clear as the C++ version -- however, in my experience, it is very easy to write unreadable expressions using the C++ syntax.

I note that in the whole of the CoCoALib source we have used the C++ ? operator about 20 times (and 7 of these are in PPMonoidEvZZ.C). So my personal experience is rather limited.

In summary do we prefer to have a fixed mapping from BOOL to INT (surely false -> 0 and true -> 1)? Or do we prefer a C/C++ approach where the programmer must specify each time the two values?

Right now I do not have a proposal for the syntax of a C++-like selector; it may be possible to devise an "extensible" syntax which can be used to offer both approaches -- or would that be overkill?

#4 - 21 May 2012 15:54 - Anna Maria Bigatti

- Category set to CoCoA-5 function: new

I don't like an automatic conversion.
We could write a conversion function **Bool01** (the shortest name I could think of) doing it.

#5 - 25 May 2012 11:41 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 80

After speaking to Anna about the various possibilities, she convinced me that the simplest approach is best (at least as a first attempt). So I approve the idea of writing a fn **Bool01** which effects the conversion.

At some later date, if there is evidence of need for a better integrated conversion scheme, we can reconsider some of my musings from earlier postings. Given how rarely we have used the C++ construct in the CoCoALib sources, it seems likely that Bool01 will not be used much.

#6 - 25 May 2012 14:17 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 80 to 100

Implemented **Bool01** in the package NotBuiltin.cpkg5.
The fn **Error** (with capital E) is now built-in rather than defined in BackwardCompatible.cpkg5 -- this makes error mesgs (provoked by Error) easier to understand.
Also made small mods to some other impls in the BackwardCompatible.cpkg5.
Added doc for Bool01.

#7 - 04 Jul 2012 09:59 - Anna Maria Bigatti

- Assignee set to John Abbott

- Target version set to CoCoA-5.0.3