Boo Optimizations

Some things that could be optimized by the boo compiler. See also the Boo Style Checker ideas.

- (this is done, see http://jira.codehaus.org/browse/BOO-137) Convert "for i in range(x)" to an optimized while loop, instead of using the rangeenumerator class. Also optimize "for item in array".
- When comparing a single variable to several different values in a if statement (if x==1...elif x==2...), that can be optimized. Integers can be optimized to a kind of jump table. Strings can be optimized by checking if string.IsInterned first.
- Convert s=="" to s.Length==0, apparently it is faster, but also warn if user did not also check that s is not null, or else add the null check too.
- If user is calling a property or method repeatedly or in a loop, suggest assigning to a temp variable first to make it faster. This could be a Boo Style Checker idea instead.
- Suggest using rawArrayIndexing for speed improvements for indexers (something[i]).
- If result of if test is always true, remove test. If block of code does nothing, remove it.
- Self-recursive tail calls. If functions ends by calling itself again, replace that with a goto back to the start of the method. See nemerle.
- Some other optimizations:
  - don't box in multidimensional array access: http://jira.codehaus.org/browse/BOO-282
  - cache external types: http://jira.codehaus.org/browse/BOO-141
  - don't create closure shared locals class when variable not accessed outside closure: http://jira.codehaus.org/browse/BOO-101

There are many optimizations the CLR runtime does itself.