Compiler optimizations

Each of these is a relatively self-contained piece of work that should dramatically improve the performance of the code generated by the Manticore compiler.

- Loop unrolling in the CFG representation. We currently perform significantly more allocation tests than are needed, particularly within what should be straightline loop block. This involves loop detection and allocation check hoisting (so there is only one per loop iteration).
- Continuation pass ([http://portal.acm.org/citation.cfm?id=507669.507639](http://portal.acm.org/citation.cfm?id=507669.507639)). We use a CPS-based IR and already have control-flow analysis information available. Implementation of this optimization involves transforming a function that takes a known continuation parameter - and thus, always return to the same location - and transform the continuation from a parameter into an explicit call.
- Better closure conversion ([http://portal.acm.org/citation.cfm?id=345099.345125](http://portal.acm.org/citation.cfm?id=345099.345125)). We do not do a very good job of sharing closures in all possible cases, and this hurts our allocation profile, particularly within tight-loop code!