Instrumentation specification

Attendees:
Drew Bernat, Markus Geimer,
Kevin Huck, Bill Williams

Unifying source/binary instrumenter

- Consolidate keywords
  - Add new keyword to get unique names
  - Open question: How to reference them later on?
- Code specification in a unified language?
  - E.g., based on DynC
  - Requires translation into target language for source-code instrumenter
- For efficiency, given code snippet could be “outlined” to a function, parameterized by keywords used

Filtering

- Technical issue: Integrate some of the filtering stuff into Dyninst?
  - Filter interface: Provide a starting set of points and a predicate, returns a set of points satisfying the predicate
  - Transform interface: Provide set of points and a code snippet

Miscellaneous

- Is it possible to better support Extrae?
  - Needs map of function names to numbers
  - Trivial for binary instrumenter
  - Requires to preserve state between invocations for source-code instrumenter
    - Not impossible, but complicates parallel builds