1086-68-954 **Neil D. Jones*** (neil@diku.dk). *Towards relating program transformation, incremental computation and parameterized complexity.* Preliminary report.

The aim of *partial evaluation* is automatically to speed up a program p with two inputs s and d (static and dynamic) by *specialising* p to a known value of s. Speedup owes to *removal of subcomputations depending only on s.* Typically linear speedup, with coefficient depending on the value of s, and large enough to be of practical interest.

Partial evaluators (also self-applicable) exist for programming languages Scheme, Prolog and C. Applications: optimisation; compiling and compiler generation from programming language interpreters. Emphasis is on automation (no human interaction).

Supercompilation and distillation: automatic transformations on the call-by-name lambda calculus (+ constructors and recursive function definitions). Well-quasi orders are a key technique to ensure that transformation terminates. Distillation can yield superlinear speedup, by eliminating repeated subcomputations.

Incremental computation can yield still greater speedups by reusing repeated subcomputations, but is less automatic.

Anticipated: a correspondence between "binding times" central to partial evaluation, optimisations done in incremental computation, and the order in which problem parameters are given for parameterized complexity. (Received September 17, 2012)