Quantifier Support

- Quantifiers appear often in program verification
  - loop invariants, object invariants
  - specify properties of recursive data structures
  - partial specification of uninterpreted functions and predicates

Quantifiers and Nelson-Oppen

- Deal with ∃ by introducing fresh skolem constants
- For ∀ use heuristic instantiation – introduce ground instances that suffice for deciding given problem
- A common heuristic uses pattern matching (upto equivalence), e.g., given

  $f(a)=b \quad \wedge \quad g(b) \neq f(a) \quad \wedge \quad (\forall x \,.\, f(x) = \underline{g(f(x))})$

  matching introduces the instance

  $x := a$

Challenges with Quantifiers

- Automatically inferring patterns
- Improving matching performance
  - cf. various optimizations in Simplify
- **Reducing unnecessary instantiations**
  - use SAT solver / theories to prune instantiations (Verifun / Zap)

Interface with SAT solvers

- Verifun uses the SAT solver as a black box
- Key advantage: can use current world champion
- Limitation: traditional SAT interface is too narrow
  - given a set of clauses, return an assignment
- Certain optimizations need more functionality
  - e.g., incremental solving, generation of unsat cores (zChaff now provides some support)

Areas for collaboration

- Convince SAT community to provide wider interfaces
- Build benchmark suites (SMT-LIB initiative)
- Agree on common interface to decision procedures
  - enable creation of theory libraries