6. Conclusions

- Schema frameworks place program schemas into a formal setting to give us tractable, correct generation of (constraint) programs from specifications.
- Each schema captures a family of "similar" programs.
- We have good theoretical underpinnings for our approach.
  - We add axioms to the program schemas to ensure that they have the meaning we intend.
  - The resulting specification frameworks are first-order parameterised theories, with isoinitial models.
  - Steadfastness ensures correct reusability.
- We consider frameworks for different kinds of program specifications – iff, super-and-sub, …
- If our original specifications are executable, then we are doing program transformation.

- We apply schemas in a recursive specification reduction process.
- Applying a schema reuses some existing stored programs, and reduces the specification to a number of subproblems.
- The formal properties of schema frameworks ensure correctness.
- Automation of schema-based generation has two aspects:
  - Schema application results in proof obligations we must satisfy. Schemas must be carefully designed to make these tractable.
    - Schemas have an important role in organising the domain theory so that we have the right theory to tackle subproblems.
  - To automatically select a schema to apply, we use domain-specific heuristics. These are incorporated into the schema frameworks.
    - Some heuristics from inductive proof are useful. Other examples are complexity criteria, mode information,…
- Adding state transitions makes specification frameworks into components for object-oriented design.