The HOL-OCL Vision:  
A Tool Supported Formal Model-driven Engineering Process with Tool-support
Proof Obligations: Liskov’s Substitution Principle

**Liskov substitution principle**

Let \( q(x) \) be a property provable about objects \( x \) of type \( T \). Then \( q(y) \) should be true for objects \( y \) of type \( S \) where \( S \) is a subtype of \( T \).

For constraint languages, like oo, this boils down to:
- pre-conditions of overridden methods must be weaker.
- post-conditions of overridden methods must be stronger.

Which can formally expressed as implication:
- Weakening the pre-condition:
  \[ op_{pre} \rightarrow op_{sub}^{pre} \]
- Weakening the pre-condition:
  \[ op_{sub}^{post} \rightarrow op_{post} \]

Conclusion

- We presented HOL-OCL providing:
  - a formal, machine-checked semantics for OO specifications,
  - an interactive proof environment for OO specifications,
  - next (major) release planned in November 2008.
- HOL-OCL is integrated into a toolchain providing:
  - code generators,
  - a transformation framework (including PO generation),
  - support for SecureUML via model transformations.

Methodology

A tool-supported methodology should
- integrate into existing toolchains and processes,
- provide a unified approach, integrating,
  - syntactic requirements (well-formedness checks),
  - generation of semantics requirements (proof obligations),
  - means for verification (proving) or validation, and of course
- all phases should be supported by tools.

Example

A package-based object-oriented refinement methodology.

Ongoing and Future Work

- Ongoing work includes the development of support for:
  - well-formedness-checking,
  - proof-obligation generation (Liskov, Refinement, ),
  - consistency checking,
  - Hoare-style program verification,
  - better proof automation.
- Future works could include the development for
  - integrating OCL validation tools, e.g., USE,
  - test-case generation (i.e., integrating HOL-TestGen),
  - supporting SecureUML natively.
  - ....
The next Challenge for OCL Tools

- **State of the art:**
  - There are a lot of good OCL tools, which work in isolation.
  - There is no “one sizes fits all” OCL tool.
  - There is no (integrated) development process supporting.

- **Observation:** Successful specification languages comprise:
  - tools that work together.
  - one or more development processes that are well supported by tools.

- **Conclusion:** We, as the OCL Community, should
  - combining the strenghs of different OCL tools.
  - provide methodologies (development processes) on top of OCL.