New tools in HOL 4 - Consequence Conversions, Quantifier Heuristics, Styles

Thomas Tuerk

24th November 2009

Consequence Conversions

- **conversions** are ML-functions that given a term `t` return a theorem `|- t = t_eq`.
- **consequence conversions** are ML-function that given a boolean term `t` return a theorem
  - `|- t_strong ==> t`,
  - `|- t = t_eq` or
  - `|- t ==> t_weak`.
- main function `DEPTH_CONSEQ_CONV`
- functions similar to `THENC`, `REPEATC`

Consequence Conversions II

- `DEPTH_CONSEQ_CONV` supports
  - multiple consequence conversions
  - stepping / weights
  - context
  - assumptions
  - congruences
  - cashing
- there are congruences for the standard boolean operators
- `CONSEQ_REWRITE_TAC` and similar tools provide support for rewriting with implications

Quantifier Instantiation Heuristics

- given a term `?x. P x` there are 3 reasons to instantiate `x` with a concrete value `I y`:
  1. `P i`
  2. `!i'. ~(i = i') ==> ~(P i')`
  3. `!i'. P i' ==> P i`
- dual to these reasons there are three reasons for all-quantification
- `quantHeuristicsLib` is a library that supports instantiating quantifiers based on heuristics that come up with these guesses

Quantifier Instantiation Heuristics II

- a **quantifier heuristic** is an ML-function that given a term `P x` with a free variable `x` returns a list a **guesses** on how to instantiate `x`
- a guess consists of
  - the instantiation `i`
  - a list of free variables in `i` that should remain quantified
  - one of the 6 reasons or an *I-just-feel-like-it* reason
  - possibly a justification in form of a HOL-theorem
- if a justification is given, equivalence can be proved
- otherwise an implication is introduced

Quantifier Instantiation Heuristics III

- library knows about common boolean operators
- there is support for equations
- additional heuristics, rewrites, etc. are grouped in **Q***uantifier* **H***euristic* **C***ombined* **A***rguments* (QHCAs)
- `quantHeuristicsArgsLib` contains QHCAs for
  - option types
  - pairs
  - lists
  - natural number
- there is a QHCA for type-base and a stateful one
- all default heuristics come with a justifying theorems and therefore guaranteed to preserve equality
- user heuristics can be added easily

Quantifier Instantiation Heuristics IV - Interface

- `QUANT_INSTANTIATE_CONV`
- `QUANT_INSTANTIATE_TAC`
- `FAST_QUANT_INSTANTIATE_TAC`
- `QUANT_INSTANTIATE_CONSEQ_CONV`
- `FAST_QUANT_INSTANTIATE_CONSEQ_CONV`
- `EXTENSIBLE_QUANT_INSTANTIATE_CONV`
- `QUANT_INST_TAC`

Annotations & Styles

- Michael Norish added *annotations* recently to the pretty printer
- annotations convey semantic information
- bound / free variables and types are nicely presented
- I now added *styles* to do user defined presentations
- instead of the semantics one states the desired formating
  - bold
  - underline
  - foreground colour
  - background colour