# Summary of the res_quan library

W. Wong

15 April 1993

The res_quan library privides some basic facilities for manipulate restricted quantifications. It consists of a single theory, res_quan, which contains a number of theorems about the properties of some restricted quantifiers, and a set of ML functions for dealing with them. This summary lists all theorem stored in the res_quan theory and ML functions available in the library.

## 1 The theory res_quan

This theory caontains the following theorems.

```
DISJ_RESQ_EXISTS_DIST  (res_quan)
   |- !P Q R.
        (?i :: \i. P i \/ Q i. R i) = (?i :: P. R i) \/ (?i :: Q. R i)

RESQ_EXISTS_DISJ_DIST  (res_quan)
   |- !P Q R. (?i :: P. Q i \/ R i) = (?i :: P. Q i) \/ (?i :: P. R i)

RESQ_EXISTS_REORDER  (res_quan)
   |- !P Q R. (?i :: P. ?j :: Q. R i j) = (?j :: Q. ?i :: P. R i j)

RESQ_EXISTS_UNIQUE  (res_quan)
   |- !P j. (?i :: $= j. P i) = P j

RESQ_FORALL_CONJ_DIST  (res_quan)
   |- !P Q R. (!i :: P. Q i /\ R i) = (!i :: P. Q i) /\ (!i :: P. R i)

RESQ_FORALL_DISJ_DIST  (res_quan)
   |- !P Q R.
        (!i :: \i. P i \/ Q i. R i) = (!i :: P. R i) /\ (!i :: Q. R i)

RESQ_FORALL_FORALL  (res_quan)
   |- !P R x. (!x. !i :: P. R i x) = (!i :: P. !x. R i x)
```

```
RESQ_FORALL_REORDER  (res_quan)
   |- !P Q R. (!i :: P. !j :: Q. R i j) = (!j :: Q. !i :: P. R i j)

RESQ_FORALL_UNIQUE  (res_quan)
   |- !P j. (!i :: $= j. P i) = P j
```

## 2 ML functions in the library

### Conditional rewriting tools

- `COND_REWRITE1_CONV : (thm list -> thm -> conv)`
  A simple conditional rewriting conversion.

- `COND_REWRITE1_TAC : thm_tactic`
  A simple conditional rewriting tactic.

- `COND_REWR_CANON : thm -> thm`
  Transform a theorem into a form accepted by `COND_REWR_TAC`.

- `COND_REWR_CONV`
  `: ((term -> term -> ((term # term) list # (type # type) list) list)`
  ` -> thm -> conv)`
  A lower level conversion implementing simple conditional rewriting.

- `COND_REWR_TAC`
  `: ((term -> term -> ((term # term) list # (type # type) list) list)`
  `-> thm_tactic)`
  A lower level tactic used to implement simple conditional rewriting tactic.

### Syntax functions

- `mk_resq_abstract : ((term # term # term) -> term)`
  Term constructor for restricted abstraction.

- `mk_resq_exists : ((term # term # term) -> term)`
  Term constructor for restricted existential quantification.

- `mk_resq_forall : ((term # term # term) -> term)`
  Term constructor for restricted universal quantification.

- `mk_resq_select : ((term # term # term) -> term)`
  Term constructor for restricted choice quantification.

- `list_mk_resq_exists : ((term # term) list # term) -> term)`
  Iteratively constructs a restricted existential quantification.

- `list_mk_resq_forall : ((term # term) list # term) -> term)`
  Iteratively constructs a restricted universal quantification.

- `dest_resq_abstract : (term -> (term # term # term))`
  Breaks apart a restricted abstract term into quantified variable, predicate and body.

- `dest_resq_exists : (term -> (term # term # term))`
  Breaks apart a restricted existentially quantified term into quantified variable, predicate and body.

- `dest_resq_forall : (term -> (term # term # term))`
  Breaks apart a restricted universally quantified term into quantified variable, predicate and body.

- `dest_resq_select : (term -> (term # term # term))`
  Breaks apart a restricted choice quantified term into quantified variable, predicate and body.

- `strip_resq_exists : (term -> ((term # term) list # term))`
  Iteratively breaks apart a restricted existenially quantified term.

- `strip_resq_forall : (term -> ((term # term) list # term))`
  Iteratively breaks apart a restricted universally quantified term.

- `is_resq_abstract : (term -> bool)`
  Tests a term to see if it is a restricted abstraction.

- `is_resq_exists : (term -> bool)`
  Tests a term to see if it is a restricted existential quantification.

- `is_resq_forall : (term -> bool)`
  Tests a term to see if it is a restricted universal quantification.

- `is_resq_select : (term -> bool)`
  Tests a term to see if it is a restricted choice quantification.

## Derived rules

- `RESQ_GEN : ((term # term) -> thm -> thm)`
  Generalizes the conclusion of a theorem to a restricted universal quantification.

- `RESQ_GENL : ((term # term) list -> thm -> thm)`
  Generalizes zero or more variables to restricted universal quantification in the conclusion of a theorem.

- `RESQ_GEN_ALL : (thm -> thm)`
  Generalizes the conclusion of a theorem over its own assumptions.

- `RESQ_HALF_EXISTS : (thm -> thm)`
  Strip a restricted existential quantification in the conclusion of a theorem.

- `RESQ_HALF_SPEC : (thm -> thm)`
  Strip a restricted universal quantification in the conclusion of a theorem.
- `RESQ_MATCH_MP : (thm -> thm -> thm)`
  Eliminating a restricted universal quatification with automatic matching.
- `RESQ_REWR_CANON : thm -> thm`
  Transform a theorem into a form accepted for rewriting
- `RESQ_SPEC : (term -> thm -> thm)`
  Specializes the conclusion of a restricted universally quantified theorem.
- `RESQ_SPECL : (term list -> thm -> thm)`
  Specializes zero or more variables in the conclusion of a restricted universally quantified theorem.
- `RESQ_SPEC_ALL : (thm -> thm)`
  Specializes the conclusion of a theorem with its own quantified variables.

## Conversions

- `AND_RESQ_FORALL_CONV : conv`
  Moves a restricted universal quantification out a conjunction.
- `IMP_RESQ_FORALL_CONV : conv`
  Converts an implication to a restricted universal quantification.
- `LIST_RESQ_FORALL_CONV : conv`
  Converts restricted universal quantifications iteratively to implications.
- `RESQ_EXISTS_CONV : conv`
  Converts a restricted existential quantification to a conjunction.
- `RESQ_FORALL_AND_CONV : conv`
  Splits a restricted universal quantification across a conjunction.
- `RESQ_FORALL_CONV : conv`
  Converts a restricted universal quantification to an implication.
- `RESQ_FORALL_SWAP_CONV : conv`
  Changes the order of two restricted universal quantifications.
- `RESQ_REWRITE1_CONV : thm list -> thm -> conv`
  Rewriting conversion with restricted universally quantified theorem.

## Tactics

- `RESQ_GEN_TAC : tactic`
  Strips the outermost restricted universal quantifier from the conclusion of a goal.

- `RESQ_HALF_GEN_TAC : tactic`
  Strips the outermost restricted universal quantifier from the conclusion of a goal.

- `RESQ_EXISTS_TAC : term -> tactic`
  Strips the outermost restricted extistential quantifier from the conclusion of a goal.

- `RESQ_IMP_RES_TAC : thm_tactic`
  REpeatedly resolves a restricted univerally quantified theorem with the assumptions of a goal.

- `RESQ_IMP_RES_THEN : thm_tactical`
  Resolves a restricted univerally quantified theorem with the assumptions of a goal.

- `RESQ_RES_TAC : tactic`
  Enriches assumptions by repeatedly resolving restricted universal quantifications in them against the others.

- `RESQ_RES_THEN : thm_tactic -> tactic`
  Resolves all restricted univerally quantified assumptions against other assumptions of a goal.

- `RESQ_REWRITE1_TAC : thm_tactic`
  Rewriting with restricted universally quantified theorem.

## Constant definition

- `new_binder_resq_definition : ((string # term) -> thm)`
  Declare a new binder and install a definitional axiom in the current theory.

- `new_infix_resq_definition : ((string # term) -> thm)`
  Declare a new infix constant and install a definitional axiom in the current theory.

- `new_resq_definition : ((string # term) -> thm)`
  Declare a new constant and install a definitional axiom in the current theory.