Isabelle/HOL Proof Assistant

Isar engine

Use of Natural Deduction (ND) instead of Gentzen sequent calculus (LJ/LK). Thus, Γ is a **metacontext** and its objects can be used with following commands : `assume A`, `show B`...

Syntax of Isabelle commands

Inspection Commands

- type-checking terms : `term "⟨hol-term⟩"`
- evaluating terms : `value "⟨hol-term⟩"`
- explore libraries :
  `find_theorems <theorem content>* [name: <theorem name>]*`

Specification Commands

- non-recursive definitions : `definition f :: "⟨τ⟩"`
  `where name : "f x1 ... xn = ⟨τ⟩"`
- type definitions :
  `typedef ('a1, ..., 'an) κ = "⟨set-expr⟩" ⟨proof⟩`

  Example. `typedef even = "{ x :: int. x mod 2 = 0 }"`

Specification Mechanism Commands

- datatype definitions:

  `datatype ('a1, ..., 'an) = ⟨c⟩ :: "⟨τ⟩"|...|⟨c⟩ :: "⟨τ⟩"`

- recursive function definitions :
  `typedef ('a1, ..., 'an) κ = "⟨set-expr⟩" ⟨proof⟩`

  Example. `typedef even = "{ x :: int. x mod 2 = 0 }"`

- inductively defined sets:

```
inductive <c> [ for <v> :: "<τ>" ]
  where <thmname> : "<φ>"
      | ...
      | <thmname> = "<φ>"
```

  Example.

```
inductive path for rel :: "'a ⇒ 'a ⇒ bool"
where base : "path rel x x"
    | step : "rel x y ⇒ path rel y z ⇒ path rel x z"
```

- extended notation for cartesian products (records):

```
record <c> = [<record> +]
  tag_1 :: "<τ_1>"
  ...
  tag_n :: "<τ_n>"
```

Apply rules and theorems

- `apply assumption` proves $[\![B_1, \ldots, B_m]\!] \Rightarrow C$ by unifying C with one of the B_i
- `apply (rule ⟨intro-rule⟩)` :
  1. decompose formulae to the right of $\Rightarrow$
  2. applying $[\![A_1, \ldots, A_n]\!] \Rightarrow A$ to subgoal C,
     (a) unify A and C
     (b) replace C with n new subgoals $A_1, \ldots, A_n$
- `apply (erule ⟨elim-rule⟩)` :
  1. decompose formulae to the left of $\Rightarrow$
  2. applying $[\![A_1, \ldots, A_n]\!] \Rightarrow A$ to subgoal C, like rule but also
     (a) unifies first premise of rule with an assumption
     (b) eliminates that assumption
- `apply (case_tac ⟨term⟩)` : case distinctions on arbitrary terms (e.g. `excluded_middle` on type bool)
- `apply (rule_tac x = "⟨term⟩" in ⟨rule⟩)`
  1. like rule but $?x$ in ⟨rule⟩ is instanciated by ⟨term⟩ before application
  2. applying $[\![A_1, \ldots, A_n]\!] \Rightarrow A$ to subgoal C,
     (a) x is in ⟨rule⟩, not in goal
     (b) ⟨term⟩ may contain parameters from the goal and those introduced in Isar texts
- `apply (erule_tac x = "⟨term⟩" in ⟨rule⟩)` : similar
- `apply (rename_tac` $x_1 \ldots x_n$`)` renames the rightmost (inner) n parameters to $x_1, \ldots, x_n$
- `apply (frule ⟨rule⟩)` : ?
- `apply clarify` : applies all safe rules that do not split the goal
- `apply safe` : applies all safe rules
- `apply fast` : sequent based automatic
- `apply best` : search tactics
- `apply blast` : automatic tableaux prover (works well on predicate logic)
- `apply metis` : resolution prover for FO with equality
- `insert` ?

Logical rules

Rules Safety

Definition. Safe rules preserve provability.

- safe: `conjI`, `impI`, `notI`, `iffI`, `refl`, `ccontr`, `classical`, `conjE`, `disjE`, `allI`, `exE`
- unsafe: `disjl1`, `disjl2`, `impE`, `iffD1`, `iffD2`, `notE`, `allE`, `exI`

1. Apply safe rules before unsafe ones
2. Create parameters first, unknowns later

Description operator

1. ε-operator : $\varepsilon x\ P(x)$ is a value that satisfies P (if exists) ; written in Isabelle as $\texttt{SOME}\ x.Px$ s.t.

$$\frac{P(?x)}{P(\texttt{SOME}\ x.P(x))}\ \texttt{someI}$$

2. ι-operator : ε implies Axiom of Choice[1] :

Intuitionistic logic (LJ)

Name	Rule in Gentzen style	Desc.
`TrueI`	$\dfrac{}{\Gamma \vdash \top}$	
`FalseE`	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash P}$	
`notI`	$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$	
`notE`	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash P}$	
`conjI`	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$	
`conjE`	$\dfrac{\Gamma \vdash A \wedge B \quad \Gamma, A, B \vdash C}{\Gamma \vdash C}$	
`conjunct1`	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$	
`conjunct2`	$\dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$	
`context_conjI`	$\dfrac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash A \wedge B}$	
`disjI1`	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$	
`disjI2`	$\dfrac{\Gamma \vdash A}{\Gamma \vdash B \vee A}$	
`disjCI`	$\dfrac{\Gamma, \neg B \vdash A}{\Gamma \vdash A \vee B}$	
`disjE`	$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$	
`impI`	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$	
`impE`	$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma \vdash C}$	
`impCE`	$\dfrac{\Gamma, A \vdash B \quad \Gamma, \neg A \vdash B}{\Gamma \vdash B}$	
`mp`	$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	$\Rightarrow$-elim
`iffI`	$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash B \Rightarrow A}{\Gamma \vdash A \Leftrightarrow B}$	
`iffE`	$\frac{\Gamma \vdash A \Leftrightarrow B \quad \Gamma, A \Rightarrow B, B \Rightarrow A \vdash C}{\Gamma \vdash C}$	
`iffD1`	$\dfrac{\Gamma \vdash A \Leftrightarrow B}{\Gamma \vdash A \Rightarrow B}$	
`iffD2`	$\dfrac{\Gamma \vdash A \Leftrightarrow B}{\Gamma \vdash B \Rightarrow A}$	

[1] Axiom of Choice : $\forall x\ \exists y\ Pxy \Rightarrow \exists f\ \forall x\ P\ x\ (fx)$

De Morgan Extensions

`notnotD`	$\dfrac{\Gamma, \neg\neg P}{\Gamma \vdash P}$
`de_Morgan_disj`	$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$
`de_Morgan_conj`	$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
`disj_not1`	$\neg P \vee Q \Leftrightarrow P \Rightarrow Q$
`disj_not2`	$P \vee \neg Q \Leftrightarrow Q \Rightarrow P$

Non-Constructive Classical Logic (LK)

Name	Rule in sequent style	Desc.
`True_or_false`	$\dfrac{}{\Gamma \vdash A \Leftrightarrow \top \vee A \Leftrightarrow \bot}$	
`classical`	$\dfrac{\Gamma, \neg A \vdash A}{\Gamma \vdash A}$	absurd ?
`excluded_middle`	$\dfrac{}{\Gamma \vdash A \vee \neg A}$	
`ccontr`	$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash \bot}$	

Non-Constructive First Order Logic (FO)

- $\bigwedge x$: new parameters introduced
- $?x$: new unknowns introduced

Name	Rule in sequent style	Desc.
`allI`	$\dfrac{\Gamma \vdash \bigwedge x\ P(x)}{\Gamma \vdash \forall x\ P(x}$	$\forall$-intro
`allE`	$\dfrac{\Gamma \vdash \forall x\ P \quad \Gamma, P(?x) \vdash Q}{\Gamma \vdash Q}$	
`exI`	$\dfrac{\Gamma \vdash P(?x)}{\Gamma \vdash \exists x\ P(x)}$	$\exists$-intro
`exE`	$\dfrac{\Gamma \vdash \exists x\ P \quad \Gamma, \bigwedge x\ P(x) \vdash Q}{\Gamma \vdash Q}$	$\exists$-elim
`spec`	$\dfrac{\Gamma \vdash \forall x\ P(x)}{\Gamma \vdash P(?x)}$	

Equational Logic

Name	Rule in sequent style	Desc.
`refl`	$\dfrac{}{x = x}$	
`sym`	$\dfrac{y = x}{x = y}$	
`trans`	$\dfrac{x = y \quad y = z}{x = z}$	
`subst`	$\dfrac{x = y \quad P(x)}{P(y)}$	
`ext`	$\dfrac{\wedge t\ P(t) = Q(t)}{P = Q}$	