Topics in Automated Deduction (CS 576)

Elsa L. Gunter

2112 Siebel Center

egunter@illinois.edu

http://www.cs.illinois.edu/class/sp10/cs576/

Adding Assumptions to Goals:

subgoal_tac

- Can always add assumption *asm* to current subgoal with

  `apply (subgoal_tac "`*asm*`")`

- Statement can use Isabelle parameters
- Adds new subgoal *asm* with same assumptions as current subgoal

Adding Assumptions to Goals:

subgoal_tac

---

1. $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$

```
apply (subgoal_tac "asm")
```

yields

1. $[\![A_1; \ldots; A_n; asm]\!] \Longrightarrow A$
2. $[\![A_1; \ldots; A_n]\!] \Longrightarrow asm$

Removing Assumptions: thin_tac

- Can remove unwanted assumption *asm* from current subgoal with

  `apply (thin_tac "`*asm*`")`

  $$1.\ \llbracket A_1; \ldots; A_{i-1}; A_i; A_{i+1}; \ldots; A_n \rrbracket \Longrightarrow A$$

  `apply (thin_tac "`A_i`")`

  yields

  $$1.\ \llbracket A_1; \ldots; A_{i-1}; A_{i+1}; \ldots; A_n \rrbracket \Longrightarrow A$$

"Clarifying" the Goal

- `apply (intro ... )`

  Repeatedly apply intro rules

  Example: `apply (intro allI)`

- `apply (elim ... )`

  Repeatedly apply elim rules

  Example: `apply (elim conjE)`

- `apply (clarify)`

  Repeatedly apply safe rules without splitting goal

- `apply (clarsimp simp add: ... )`

  Combination of `clarify` and `simp`

Other Automated Proof Methods

- `blast` Isabelle's most powerful classical reasoner.

  Useful for goals stated using only predicate logic and set theory

  Can be extended with rules (with `[iff]` attribute) to handler broader classes of goals

- `auto`

  Applies to all subgoals. Combines classical reasoning with simplification Does what it can; leaves

unfinished subgoals Splits subgoals

Other Automated Proof Methods

- `blast`
- `force`

  Similar to `auto`, but only applies to one goal, and either finishes or fails.
- `safe`

  Like `clarify` but also splits goals

Sets

Type `'a set`: sets over type `'a`

- $\{\ \}$, $\{e_1, \ldots, e_n\}$, {x. Px}
- $e \in A$, $A \subseteq B$ (ascii: $e : A$, A <= B)
- $A \cup B$, $A \cap B$, $A - B$, $-A$
- $\bigcup_{x \in A} B\ x$, $\bigcap_{x \in A} B\ x$
- $\{i. .j\}$
- insert ::′ a ⇒′ a set ⇒′ a set
- f ‘ $A \equiv \{y.\ \exists x \in A.\ y = f\ x\}$
- ...

Proofs about Sets

Natural deduction proof rules:

- `equalityI`: $[\![A \subseteq B;\ B \subseteq A]\!] \Longrightarrow A = B$
- `equalityE`: $[\![A = B;\ [\![A \subseteq B;\ B \subseteq A]\!] \Longrightarrow P]\!] \Longrightarrow P$
- `subsetI`: $(\bigwedge x.\ x \in A \Longrightarrow x \in B) \Longrightarrow A \subseteq B$
- `sutsetD`: $[\![A \subset B;\ c \in A]\!] \Longrightarrow c \in B$

Proofs about Sets

More natural deduction proof rules:

- IntI: $[\![c \in A;\ c \in B]\!] \Longrightarrow c \in A \cap B$
- IntD1: $c \in A \cap B \Longrightarrow c \in A$
- IntD2: $c \in A \cap B \Longrightarrow c \in B$
- set_ext: $(\Lambda x.\ (x \in A) = (x \in B)) \Longrightarrow A = B$
- . . . (see Manual)

Bounded Quantification

- $\forall x \in A.\ P\ x \equiv \forall x.\ x \in A \longrightarrow P\ x$
- $\exists x \in A.\ P\ x \equiv \exists x.\ x \in A \wedge P\ x$

- ballI: $(\lambda x.x \in A \Longrightarrow P\ x) \Longrightarrow \forall x \in A.\ P\ x$
- bspec: $[\![\forall x \in A.\ P\ x;\ x \in A]\!] \Longrightarrow P\ x$
- bexI: $[\![P\ x;\ x \in A]\!] \Longrightarrow \exists x \in A.\ P\ x$
- bexE: $[\![\exists x \in A.\ P\ x;\ \Lambda x.[\![x \in A;\ P\ x]\!] \Longrightarrow Q]\!] \Longrightarrow Q$

Demo: Some Set Theory

Format for Inductive Set Definitions

$$\texttt{inductive\_set}\ \mathtt{S} :: \text{“}\tau\ \texttt{set”}\ \texttt{where}$$

$$[\![\mathtt{a}_{1,1} \in \mathtt{S}; \ldots;\ \mathtt{a}_{1,n} \in \mathtt{S}; \mathtt{A}_{1,1};\ \ldots;\ \mathtt{A}_{1,k}]\!] \Longrightarrow \mathtt{a}_1 \in \mathtt{S}\ |$$

$$\ldots\ |$$

$$[\![\mathtt{a}_{m,1} \in \mathtt{S}; \ldots;\ \mathtt{a}_{m,l} \in \mathtt{S}; \mathtt{A}_{m,1};\ \ldots;\ \mathtt{A}_{m,j}]\!] \Longrightarrow \mathtt{a}_m \in \mathtt{S}$$

where $\mathtt{A}_{i,j}$ are side conditions not involving $\mathtt{S}$.

Example: Finite Sets

Informally

- The empty set is finite
- Adding an element to a finite set yields a finite set
- These are the only finite sets

Example: Finite Sets

In Isabelle/HOL:

```
inductive_set Finites ::  'a set set
```

– The set of all finite sets

```
{ } ∈ Finites |
A ∈ Finites ⟹ insert a A ∈ Finites
```

Example: Even Numbers

Informally

- 0 is even
- If `n` is even, then so is $\texttt{n} + 2$
- These are the only even numbers

Example: Even Numbers

In Isabelle/HOL:

```
inductive_set Ev ::  nat set
```

— The set of all even numbers

$0 \in \mathtt{Ev}$ |

$\mathtt{n} \in \mathtt{Ev} \Longrightarrow \mathtt{n} + 2 \in \mathtt{Ev}$

Proving Properties of Even Numbers

Easy: $4 \in \mathtt{Ev}$

$$0 \in \mathtt{Ev} \Longrightarrow 2 \in \mathtt{Ev} \Longrightarrow 4 \in \mathtt{Ev}$$

Trickier: $\mathtt{m} \in \mathtt{Ev} \Longrightarrow \mathtt{m} + \mathtt{m} \in \mathtt{Ev}$

Idea: induct on the length of the derivation of $\mathtt{m} \in \mathtt{EV}$

Better: induct on the *structure* of the derivation

Proving Properties of Even Numbers

Induction leads to two cases:

- **rule:** $0 \in \mathtt{Ev}$

  1. $0 + 0 \in \mathtt{Ev}$ case $\mathtt{m} = 0$

- **rule:** $\mathtt{n} \in \mathtt{Ev} \Longrightarrow \mathtt{n} + 2 \in \mathtt{Ev}$

  2. $\bigwedge \mathtt{n}.[\![\mathtt{n} \in \mathtt{Ev};\ \mathtt{n}+\mathtt{n} \in \mathtt{Ev}]\!] \Longrightarrow \mathtt{Suc(Sucn)}+\mathtt{Suc(Sucn)} \in \mathtt{Ev}$

     case $\mathtt{m} = \mathtt{n} + 2$

Rule Induction for Ev

To prove

$$n \in Ev \Longrightarrow P\ n$$

by textitrule induction on n ∈ Ev we must prove

- P 0
- $P\ n \Longrightarrow P(n+2)$

Uses rule Ev.induct:

$$[\![\mathtt{n} \in \mathtt{Ev};\ \mathtt{P\ 0};\ \mathtt{\Lambda n.\ P\ n} \Longrightarrow \mathtt{P(n+2)}]\!] \Longrightarrow \mathtt{P\ n}$$

An elimination rule

Rule Induction in General

Set S is defined inductively. To prove

$$x \in S \Longrightarrow P\ x$$

by *rule induction* on $x \in S$ we must prove for every rule

$$[\![a_1 \in S;\ \ldots;\ a_n \in S]\!] \Longrightarrow a \in S$$

that P is preserved:

$$[\![P\ a_1;\ \ldots;\ P\ a_n]\!] \Longrightarrow P\ a$$

In Isabelle/HOL:

`apply(erule S.induct)`

Demo: Inductive Set Definition

Demo: Evens are infinite

Format for Inductive Relations Definitions

inductive R :: "$\tau \longrightarrow$ bool" where

$$[\![R(a_{1,1}); \ldots; R(a_{1,n}); A_{1,1}; \ldots; A_{1,k}]\!] \Longrightarrow R(a_1) \mid$$

$$\ldots \mid$$

$$[\![R(a_{m,1}); \ldots; R(a_{m,l}); A_{m,1}; \ldots; A_{m,j}]\!] \Longrightarrow R(a_m)$$

where $A_{i,j}$ are side conditions not involving R.