Automated Reasoning

Lecture 4: Propositional Reasoning in Isabelle

Jacques Fleuriot
jdf@inf.ed.ac.uk
Recap

Last lecture:
- Completed the natural deduction system for propositional logic
- Started on proving propositions in Isabelle

Today:
- More details on proving propositions in Isabelle
- Alternative inference rules (L-system, a.k.a. “Sequent Calculus”)
- Why should we trust Isabelle?
The rule Method

To apply an inference rule, we use rule.

Consider the theorem disjI1

\[ ?P \implies ?P \lor ?Q \]

Using the command

```
apply (rule disjI1)
```

on the goal

\[ [A; B; C] \implies (A \land B) \lor D \]

yields the subgoal

\[ [A; B; C] \implies A \land B \]
General definition of method rule

When we apply the method rule someRule where

\[
someRule : [P_1; \ldots; P_m] \rightarrow Q
\]

to the goal

\[
[A_1; \ldots; A_n] \rightarrow C
\]

where \( Q \) and \( C \) can be unified, we generate the goals

\[
[A'_1; \ldots; A'_n] \rightarrow P'_1
\]

\[
\vdots
\]

\[
[A'_1; \ldots; A'_n] \rightarrow P'_m
\]

where \( A'_1, A'_2, \ldots, A'_n, P'_1, P'_2, \ldots, P'_m \) are the results of applying the substitution which unifies \( Q \) and \( C \) to \( A_1, A_2, \ldots, A_n, P_1, P_2, \ldots, P_m \).

We must now derive each of the rule’s assumptions using our goal’s assumptions.
A Problem with rule

Consider the \texttt{disjE} rule:

\[
\text{disjE} : [P \lor Q; P \rightarrow R; Q \rightarrow R] \rightarrow R
\]

If we have the goal:

\[
[(A \land B) \lor C; D] \rightarrow B \lor C
\]

Then applying rule \texttt{disjE} produces three new goals:

\[
[(A \land B) \lor C; D] \rightarrow ?P \lor ?Q
\]

\[
[(A \land B) \lor C; D; ?P] \rightarrow B \lor C
\]

\[
[(A \land B) \lor C; D; ?Q] \rightarrow B \lor C
\]

We then solve the first subgoal by applying assumption.

This seems pointlessly roundabout... we often want to use one of our assumptions in our proof.
The erule Method

Used when the conclusion of theorem matches the conclusion of the current goal and the first premise of theorem matches a premise of the current goal.

Consider the theorem **disjE**

\[ [P \lor Q; P \implies R; Q \implies R] \implies R \]

Applying erule **disjE** to goal

\[ [(A \land B) \lor C; D] \implies B \lor C \]

yields the subgoals

\[ [D; (A \land B)] \implies B \lor C \quad [D; C] \implies B \lor C \]
General definition of method erule

When we apply the method erule someRule where

\[ \text{someRule}: [P_1; \ldots; P_m] \Rightarrow Q \]

to the goal

\[ [A_1; \ldots; A_n] \Rightarrow C \]

where \( P_1 \) and \( A_1 \) are unifiable and \( Q \) and \( C \) are unifiable, we generate the goals:

\[ [A_2'; \ldots; A_n'] \Rightarrow P_2' \]

\[ \vdots \]

\[ [A_2'; \ldots; A_n'] \Rightarrow P_m' \]

where \( A_2', \ldots, A_n', P_2', \ldots, P_m' \) are the results of applying the substitution which unifies \( P_1 \) to \( A_1 \) and \( Q \) to \( C \) to \( A_2, \ldots, A_n, P_2, \ldots, P_m \).

We eliminate an assumption from the rule and the goal, and must derive the rule’s other assumptions using our goal’s other assumptions.
General definition of method \texttt{drule}

When we apply the method \texttt{drule} \texttt{someRule} where

\[ \texttt{someRule} : [P_1; \ldots; P_m] \rightarrow Q \]

to the goal

\[ [A_1; \ldots; A_n] \rightarrow C \]

where \( P_1 \) and \( A_1 \) are unifiable, we generate the goals:

\[ [A'_2; \ldots; A'_n] \rightarrow P'_2 \]

\[ \vdots \]

\[ [A'_2; \ldots; A'_n] \rightarrow P'_m \]

\[ [Q'; A'_2; \ldots; A'_n] \rightarrow C' \]

where \( A'_2, A'_3, \ldots, A'_n, P'_2, P'_3 \ldots, P'_m, Q', C' \) are the results of applying the substitution which unifies \( P_1 \) and \( A_1 \) to \( A_2, A_3, \ldots, A_n, P_2, P_3, \ldots, P_m, Q, C \).

We delete an assumption, replacing it with the conclusion of the rule.
General definition of method frule

When we apply the method frule someRule where

\[
\text{someRule: } [P_1; \ldots; P_m] \Rightarrow Q
\]

to the goal

\[
[A_1; \ldots; A_n] \Rightarrow C
\]

where \( P_1 \) and \( A_1 \) are unifiable, we generate the goals:

\[
[A'_1; A'_2; \ldots; A'_n] \Rightarrow P'_2
\]

\[
\vdots
\]

\[
[A'_1; A'_2; \ldots; A'_n] \Rightarrow P'_m
\]

\[
[Q'; A'_1; A'_2; \ldots; A'_n] \Rightarrow C'
\]

where \( A'_1, A'_2, \ldots, A'_n, P'_2, \ldots, P'_m, Q', C' \) are the results of applying the substitution which unifies \( P_1 \) and \( A_1 \) to \( A_1, A_2, \ldots, A_n, P_2, \ldots, P_m, Q, C \).

This is like drule except the assumption in our goal is kept.
More Methods

- rule_tac, erule_tac, drule_tac and frule_tac are like their counterparts, but you can give substitutions for variables in the rule before they are applied.

**Example**

```
apply (erule_tac Q="B ∧ D" in conjE)
```

applied to the subgoal

\[[A ∧ B; C ∧ B ∧ D] \implies B ∧ D\]

generates the new goal

\[[A ∧ B; C; B ∧ D] \implies B ∧ D\]

- Isabelle also provides advanced tactics, like simp and auto which perform some automatic deduction.
The `erule` tactic points to another way of phrasing a system of inference rules in a system with sequents $\Gamma \vdash A$.

Instead of elimination rules:

- $\Gamma \vdash P \lor Q$
- $\Gamma, P \vdash R$
- $\Gamma, Q \vdash R$

$\Gamma \vdash R$ (disjE)

Have left introduction rules (all the introduction rules in natural deduction introduce connectives on the right-hand side of the $\vdash$):

- $\Gamma, P \vdash R$
- $\Gamma, Q \vdash R$

$\Gamma, P \lor Q \vdash R$

This corresponds to applying rules using `erule` in Isabelle.

The left introduction rules are often much easier to use in a backwards, goal-directed style.
The following \( L \)-System (a.k.a. Sequent Calculus) rules are an alternative sound and complete proof system for propositional logic:

\[
\begin{align*}
\Gamma \vdash P & \quad \text{(assumption)} \\
\Gamma \vdash P, Q & \quad \text{(conjI)} \quad \Gamma, P, Q \vdash R \quad \text{(e conjE)} \\
\Gamma \vdash P & \quad \Gamma \vdash P \lor Q \quad \text{(disjI1)} \quad \Gamma \vdash Q & \quad \Gamma \vdash P \lor Q \quad \text{(disjI2)} \quad \Gamma, P \vdash R & \quad \Gamma, Q \vdash R \quad \text{(e disjE)} \\
\Gamma, A \vdash B & \quad \Gamma \vdash A \rightarrow B \quad \text{(impI)} \quad \Gamma \vdash P & \quad \Gamma, Q \vdash R \quad \text{(e impE)} \quad \Gamma, \bot \vdash P & \quad \text{(e FalseE)} \\
\Gamma, P \vdash \bot & \quad \Gamma \vdash \neg P \quad \text{(notI)} \quad \Gamma, P, \neg P \vdash R & \quad \text{(e notE)} \quad \Gamma \vdash P \lor \neg P & \quad \text{(excluded_middle)}
\end{align*}
\]

Note: \( e \) someRule is short for erule someRule.

Note: in the above presentation left-hand-sides are sets of formulas.
An Old Friend Revisited

\[
\begin{align*}
\frac{S, \neg S \vdash R}{S \vdash R} & \quad \text{(e notE)} \\
\frac{R, \neg S \vdash R}{(S \lor R), \neg S \vdash R} & \quad \text{(assumption)} \\
\frac{(S \lor R), \neg S \vdash R}{(S \lor R) \land \neg S \vdash R} & \quad \text{(e disjE)} \\
\frac{(S \lor R) \land \neg S \vdash R}{\vdash (S \lor R) \land \neg S \rightarrow R} & \quad \text{(e ConjE)}
\end{align*}
\]
Re-using proofs: The Cut rule

So far, all proofs have been self-contained; they have only used the pre-existing rules of inference.

By the completeness theorem, this suffices to prove everything that is true, but can lead to extremely repetitive proofs.
Re-using proofs: The Cut rule

So far, all proofs have been self-contained; they have only used the pre-existing rules of inference.

By the completeness theorem, this suffices to prove everything that is true, but can lead to extremely repetitive proofs.

The cut rule: (we “cut” $P$ into the proof)

$$
\begin{array}{c}
\Gamma \vdash P \\
\Gamma, P \vdash Q \\
\Gamma \vdash Q
\end{array}
$$

allows the use of a lemma $P$ in a proof of $Q$. We can now reuse $P$ multiple times in the proof of $Q$. 
Re-using proofs: The Cut rule

So far, all proofs have been self-contained; they have only used the pre-existing rules of inference.

By the completeness theorem, this suffices to prove everything that is true, but can lead to extremely repetitive proofs.

The cut rule: (we “cut” $P$ into the proof)

\[
\frac{\Gamma \vdash P \quad \Gamma, P \vdash Q}{\Gamma \vdash Q}
\]

allows the use of a lemma $P$ in a proof of $Q$. We can now reuse $P$ multiple times in the proof of $Q$.

In Isabelle:
- `cut_tac lemmaName` — adds the conclusion of `lemmaName` as a new assumption, and its assumptions as new subgoals
- `subgoal_tac P` — adds $P$ as a new assumption, and introduces $P$ as a new subgoal.
Why should you believe Isabelle?

When Isabelle says “No subgoals!” why should we believe that we have really proved something? Is Isabelle sound?

It is doing non-trivial work behind the scenes: unification, rewriting, maintaining a database of theorems+assumptions, automatic proof.

---

Isabelle uses two strategies to maintain soundness:

▶ A small trusted kernel: internally, every proof is broken down into primitive rule applications which are checked by a small piece of hand-verified code. This is the “LCF” model. So new proof procedures cannot introduce unsoundness.

▶ Encourages definitional extension of the logic: new concepts are introduced by definition rather than axiomatisation (more on this in Lecture 6). So new definitions cannot introduce unsoundness.

Threats to (practical) soundness still exist, including: Have we proved what we thought we proved? Are the formulas displayed on screen correctly? …

Why should you believe Isabelle?

When Isabelle says “No subgoals!” why should we believe that we have really proved something? Is Isabelle sound?

It is doing non-trivial work behind the scenes: unification, rewriting, maintaining a database of theorems+assumptions, automatic proof.

Isabelle uses two strategies to maintain soundness:

- **A small trusted kernel**: internally, every proof is broken down into primitive rule applications which are checked by a small piece of hand-verified code. This is the “LCF” model. So new *proof procedures* cannot introduce unsoundness.
- **Encourages definitional extension of the logic**: new concepts are introduced by definition rather than axiomatisation (more on this in Lecture 6). So new definitions cannot introduce unsoundness.
Why should you believe Isabelle?

When Isabelle says “No subgoals!” why should we believe that we have really proved something? Is Isabelle sound?

It is doing non-trivial work behind the scenes: unification, rewriting, maintaining a database of theorems+assumptions, automatic proof.

Isabelle uses two strategies to maintain soundness:

▶ A small trusted kernel: internally, every proof is broken down into primitive rule applications which are checked by a small piece of hand-verified code. This is the “LCF” model. So new proof procedures cannot introduce unsoundness.

▶ Encourages definitional extension of the logic: new concepts are introduced by definition rather than axiomatisation (more on this in Lecture 6). So new definitions cannot introduce unsoundness.

Threats to (practical) soundness still exist, including: Have we proved what we thought we proved? Are the formulas displayed on screen correctly? ...

Summary

- More tools for proving propositions in Isabelle
  - The erule, drule, frule methods
  - Their -tac variants
  - L-systems, and Cut rules (cut_tac, subgoal_tac).
  - See the Prop.thy file on the webpage and start using Isabelle (if you haven’t done so already).

- How Isabelle maintains soundness
  - Small trusted kernel
  - Definitional extension instead of axiomatic extension

- Next time:
  - First-Order Logic: $\forall x.P$ and $\exists x.P$