NICTA

COMP 4161

NICTA Advanced Course

Advanced Topics in Software Verification

Simon Winwood, Toby Murray, June Andronick, Gerwin Klein

$\lambda^{\rightarrow}$ and **HOL**

Types and Terms in Isabelle

Types: $\tau \quad ::= \quad \mathsf{b} \mid {}'\nu \mid {}'\nu :: C \mid \tau \Rightarrow \tau \mid (\tau, \ldots, \tau)\ K$

$\mathsf{b} \in \{\texttt{bool}, \texttt{int}, \ldots\}$ base types

$\nu \in \{\alpha, \beta, \ldots\}$ type variables

$K \in \{\texttt{set}, \texttt{list}, \ldots\}$ type constructors

$C \in \{\texttt{order}, \texttt{linord}, \ldots\}$ type classes

Terms: $t \quad ::= \quad v \mid c \mid ?v \mid (t\ t) \mid (\lambda x.\ t)$

$v, x \in V, \quad c \in C, \quad V, C$ sets of names

Types and Terms in Isabelle

Types: $\tau \quad ::= \quad \mathsf{b} \mid {}'\nu \mid {}'\nu :: C \mid \tau \Rightarrow \tau \mid (\tau, \ldots, \tau)\ K$

$\mathsf{b} \in \{\texttt{bool}, \texttt{int}, \ldots\}$ base types

$\nu \in \{\alpha, \beta, \ldots\}$ type variables

$K \in \{\texttt{set}, \texttt{list}, \ldots\}$ type constructors

$C \in \{\texttt{order}, \texttt{linord}, \ldots\}$ type classes

Terms: $t \quad ::= \quad v \mid c \mid ?v \mid (t\ t) \mid (\lambda x.\ t)$

$v, x \in V, \quad c \in C, \quad V, C$ sets of names

➜ **type constructors**: construct a new type out of a parameter type.
Example: `int list`

Types and Terms in Isabelle

Types: $\tau \quad ::= \quad \mathsf{b} \mid {}'\nu \mid {}'\nu :: C \mid \tau \Rightarrow \tau \mid (\tau, \ldots, \tau)\ K$

$\mathsf{b} \in \{\texttt{bool}, \texttt{int}, \ldots\}$ base types

$\nu \in \{\alpha, \beta, \ldots\}$ type variables

$K \in \{\texttt{set}, \texttt{list}, \ldots\}$ type constructors

$C \in \{\texttt{order}, \texttt{linord}, \ldots\}$ type classes

Terms: $t \quad ::= \quad v \mid c \mid ?v \mid (t\ t) \mid (\lambda x.\ t)$

$v, x \in V, \quad c \in C, \quad V, C$ sets of names

➜ **type constructors**: construct a new type out of a parameter type.
Example: `int list`

➜ **type classes**: restrict type variables to a class defined by axioms.
Example: $\alpha :: order$

Types and Terms in Isabelle

Types: $\tau \quad ::= \quad \mathsf{b} \mid {}'\nu \mid {}'\nu :: C \mid \tau \Rightarrow \tau \mid (\tau, \dots, \tau)\ K$

$\mathsf{b} \in \{\texttt{bool}, \texttt{int}, \dots\}$ base types

$\nu \in \{\alpha, \beta, \dots\}$ type variables

$K \in \{\texttt{set}, \texttt{list}, \dots\}$ type constructors

$C \in \{\texttt{order}, \texttt{linord}, \dots\}$ type classes

Terms: $t \quad ::= \quad v \mid c \mid ?v \mid (t\ t) \mid (\lambda x.\ t)$

$v, x \in V, \quad c \in C, \quad V, C$ sets of names

- **type constructors**: construct a new type out of a parameter type.
  Example: `int list`
- **type classes**: restrict type variables to a class defined by axioms.
  Example: $\alpha :: order$
- **schematic variables**: variables that can be instantiated.

Type Classes

➜ similar to Haskell's type classes, but with semantic properties

axclass order $<$ ord

order_refl: "$x \le x$"

order_trans: "$[\![x \le y; y \le z]\!] \Longrightarrow x \le z$"

$\ldots$

Type Classes

➜ similar to Haskell's type classes, but with semantic properties

axclass order $<$ ord

order_refl: "$x \leq x$"

order_trans: "$[\![x \leq y; y \leq z]\!] \Longrightarrow x \leq z$"

$\ldots$

➜ theorems can be proved in the abstract

lemma order_less_trans: "$\bigwedge x ::' a :: order.\ [\![x < y; y < z]\!] \Longrightarrow x < z$"

Type Classes

- ➜ similar to Haskell's type classes, but with semantic properties

  axclass order $<$ ord

  order_refl: "$x \le x$"

  order_trans: "$\llbracket x \le y; y \le z \rrbracket \Longrightarrow x \le z$"

  $\ldots$

- ➜ theorems can be proved in the abstract

  lemma order_less_trans: "$\bigwedge x ::' a :: order.\ \llbracket x < y; y < z \rrbracket \Longrightarrow x < z$"

- ➜ can be used for subtyping

  axclass linorder $<$ order

  linorder_linear: "$x \le y \vee y \le x$"

Type Classes

NICTA

- similar to Haskell's type classes, but with semantic properties

  axclass order $<$ ord

  order_refl: "$x \le x$"

  order_trans: "$\llbracket x \le y; y \le z \rrbracket \Longrightarrow x \le z$"

  ...

- theorems can be proved in the abstract

  lemma order_less_trans: "$\bigwedge x ::' a :: order.\ \llbracket x < y; y < z \rrbracket \Longrightarrow x < z$"

- can be used for subtyping

  axclass linorder $<$ order

  linorder_linear: "$x \le y \vee y \le x$"

- can be instantiated

  instance nat :: "{order, linorder}" **by** ...

Schematic Variables

$$\frac{X \quad Y}{X \wedge Y}$$

➜ X and Y must be **instantiated** to apply the rule

Schematic Variables

$$\frac{X \quad Y}{X \wedge Y}$$

➜ X and Y must be **instantiated** to apply the rule

But: **lemma** "$x + 0 = 0 + x$"

➜ x is free

➜ convention: lemma must be true for all x

➜ **during the proof**, x must **not** be instantiated

Schematic Variables

$$\frac{X \quad Y}{X \wedge Y}$$

- ➜ X and Y must be **instantiated** to apply the rule

But: **lemma** "$x + 0 = 0 + x$"

- ➜ x is free
- ➜ convention: lemma must be true for all x
- ➜ **during the proof**, x must **not** be instantiated

Solution:

Isabelle has **free** (x), **bound** (x), and **schematic** (?X) variables.

Only schematic variables can be instantiated.

Free converted into schematic after proof is finished.

Higher Order Unification

Unification:
Find substitution σ on variables for terms s, t such that $\sigma(s) = \sigma(t)$

Higher Order Unification

Unification:
Find substitution σ on variables for terms s, t such that $\sigma(s) = \sigma(t)$

In Isabelle:
Find substitution σ on schematic variables such that $\sigma(s) =_{\alpha\beta\eta} \sigma(t)$

Higher Order Unification

Unification:
Find substitution σ on variables for terms s, t such that $\sigma(s) = \sigma(t)$

In Isabelle:
Find substitution σ on schematic variables such that $\sigma(s) =_{\alpha\beta\eta} \sigma(t)$

Examples:

$$
\begin{array}{lcl}
?X \wedge ?Y & =_{\alpha\beta\eta} & x \wedge x \\
?P\ x & =_{\alpha\beta\eta} & x \wedge x \\
P\ (?f\ x) & =_{\alpha\beta\eta} & ?Y\ x
\end{array}
$$

Higher Order Unification

Unification:
Find substitution σ on variables for terms s, t such that $\sigma(s) = \sigma(t)$

In Isabelle:
Find substitution σ on schematic variables such that $\sigma(s) =_{\alpha\beta\eta} \sigma(t)$

Examples:

$$\begin{array}{llll} ?X \wedge ?Y & =_{\alpha\beta\eta} & x \wedge x & [?X \leftarrow x, ?Y \leftarrow x] \\ ?P\ x & =_{\alpha\beta\eta} & x \wedge x & [?P \leftarrow \lambda x.\ x \wedge x] \\ P\ (?f\ x) & =_{\alpha\beta\eta} & ?Y\ x & [?f \leftarrow \lambda x.\ x, ?Y \leftarrow P] \end{array}$$

Higher Order: schematic variables can be functions.

Higher Order Unification

NICTA

➜ Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable

Higher Order Unification

- ➜ Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable
- ➜ Unification modulo $\alpha\beta\eta$ is undecidable

Higher Order Unification

- ➜ Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable
- ➜ Unification modulo $\alpha\beta\eta$ is undecidable
- ➜ Higher Order Unification has possibly infinitely many solutions

Higher Order Unification

- Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable
- Unification modulo $\alpha\beta\eta$ is undecidable
- Higher Order Unification has possibly infinitely many solutions

But:

- Most cases are well-behaved

Higher Order Unification

- ➜ Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable
- ➜ Unification modulo $\alpha\beta\eta$ is undecidable
- ➜ Higher Order Unification has possibly infinitely many solutions

But:

- ➜ Most cases are well-behaved
- ➜ Important fragments (like Higher Order Patterns) are decidable

Higher Order Unification

NICTA

- Unification modulo $\alpha\beta$ (Higher Order Unification) is semi-decidable
- Unification modulo $\alpha\beta\eta$ is undecidable
- Higher Order Unification has possibly infinitely many solutions

But:

- Most cases are well-behaved
- Important fragments (like Higher Order Patterns) are decidable

Higher Order Pattern:

- is a term in β normal form where
- each occurrence of a schematic variable is of the from $?f\ t_1\ \ldots\ t_n$
- and the $t_1\ \ldots\ t_n$ are η-convertible into n distinct bound variables

We have learned so far...

➜ Simply typed lambda calculus: $\lambda^{\rightarrow}$

We have learned so far...

- Simply typed lambda calculus: $\lambda^{\rightarrow}$
- Typing rules for $\lambda^{\rightarrow}$, type variables, type contexts

We have learned so far...

- ➜ Simply typed lambda calculus: $\lambda^{\rightarrow}$
- ➜ Typing rules for $\lambda^{\rightarrow}$, type variables, type contexts
- ➜ β-reduction in $\lambda^{\rightarrow}$ satisfies subject reduction

We have learned so far...

- Simply typed lambda calculus: $\lambda^{\rightarrow}$
- Typing rules for $\lambda^{\rightarrow}$, type variables, type contexts
- β-reduction in $\lambda^{\rightarrow}$ satisfies subject reduction
- β-reduction in $\lambda^{\rightarrow}$ always terminates

We have learned so far...

- Simply typed lambda calculus: $\lambda^{\to}$
- Typing rules for $\lambda^{\to}$, type variables, type contexts
- β-reduction in $\lambda^{\to}$ satisfies subject reduction
- β-reduction in $\lambda^{\to}$ always terminates
- Types and terms in Isabelle

Preview: Proofs in Isabelle

General schema:

lemma name: ”<goal>”
apply <method>
apply <method>
...
done

General schema:

lemma name: ”$<$goal$>$”
apply $<$method$>$
apply $<$method$>$
...
done

➜ Sequential application of methods until all **subgoals** are solved.

The Proof State

1. $\bigwedge x_1 \ldots x_p . [\![A_1; \ldots; A_n]\!] \Longrightarrow B$
2. $\bigwedge y_1 \ldots y_q . [\![C_1; \ldots; C_m]\!] \Longrightarrow D$

The Proof State

NICTA

1. $\bigwedge x_1 \dots x_p. [\![A_1; \dots ; A_n]\!] \Longrightarrow B$

2. $\bigwedge y_1 \dots y_q. [\![C_1; \dots ; C_m]\!] \Longrightarrow D$

$x_1 \dots x_p$	Parameters
$A_1 \dots A_n$	Local assumptions
B	Actual (sub)goal

Isabelle Theories

Syntax:

`theory` $MyTh$

`imports` $ImpTh_1 \ldots ImpTh_n$

`begin`

(declarations, definitions, theorems, proofs, ...)*

`end`

- ➜ $MyTh$: name of theory. Must live in file $MyTh$`.thy`
- ➜ $ImpTh_i$: name of *imported* theories. Import transitive.

Isabelle Theories

Syntax:

`theory` $MyTh$

`imports` $ImpTh_1 \ldots ImpTh_n$

`begin`

(declarations, definitions, theorems, proofs, ...)*

`end`

- ➜ $MyTh$: name of theory. Must live in file $MyTh$`.thy`
- ➜ $ImpTh_i$: name of *imported* theories. Import transitive.

Unless you need something special:

`theory` $MyTh$ `imports Main begin` $\ldots$ `end`

Natural Deduction Rules

$$\frac{}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B}{C}\ \text{conjE}$$

$$\frac{}{A \vee B}\ \frac{}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B}{C}\ \text{disjE}$$

$$\frac{}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B}{C}\ \text{conjE}$$

$$\frac{}{A \vee B} \quad \frac{}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B}{C}\ \text{disjE}$$

$$\frac{}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B}{C}\ \text{impE}$$

For each connective ($\wedge$, $\vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{}{A \vee B} \quad \frac{}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B}{C}\ \text{disjE}$$

$$\frac{}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B \qquad\qquad\qquad\qquad}{C}\ \text{disjE}$$

$$\frac{}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B \qquad\qquad}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \text{disjE}$$

$$\frac{}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{A}{A \vee B} \quad \frac{B}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \text{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Natural Deduction Rules

$$\frac{A \quad B}{A \wedge B}\ \text{conjI} \qquad \frac{A \wedge B \quad [\![A; B]\!] \Longrightarrow C}{C}\ \text{conjE}$$

$$\frac{A}{A \vee B}\ \frac{B}{A \vee B}\ \text{disjI1/2} \qquad \frac{A \vee B \quad A \Longrightarrow C \quad B \Longrightarrow C}{C}\ \text{disjE}$$

$$\frac{A \Longrightarrow B}{A \longrightarrow B}\ \text{impI} \qquad \frac{A \longrightarrow B \quad A \quad B \Longrightarrow C}{C}\ \text{impE}$$

For each connective ($\wedge, \vee$, etc):

introduction and **elemination** rules

Proof by assumption

apply assumption

proves

1. $[\![B_1; \ldots; B_m]\!] \Longrightarrow C$

by unifying C with one of the B_i

apply assumption

proves

1. $[\![B_1; \ldots; B_m]\!] \Longrightarrow C$

by unifying C with one of the B_i

There may be more than one matching B_i and multiple unifiers.

Backtracking!

Explicit backtracking command: **back**

Intro rules

Intro rules decompose formulae to the right of $\Longrightarrow$.

$$\textbf{apply}\ \text{(rule} <\text{intro-rule}>\text{)}$$

Intro rules

Intro rules decompose formulae to the right of $\Longrightarrow$.

apply (rule $<$intro-rule$>$)

Intro rule $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ means

- ➜ To prove A it suffices to show $A_1 \ldots A_n$

Intro rules

Intro rules decompose formulae to the right of $\Longrightarrow$.

apply (rule <intro-rule>)

Intro rule $[\![A_1;\ldots;A_n]\!] \Longrightarrow A$ means

- To prove A it suffices to show $A_1 \ldots A_n$

Applying rule $[\![A_1;\ldots;A_n]\!] \Longrightarrow A$ to subgoal C:

- unify A and C
- replace C with n new subgoals $A_1 \ldots A_n$

Elim rules

NICTA

Elim rules decompose formulae on the left of $\Longrightarrow$.

apply (erule <elim-rule>)

Elim rules

Elim rules decompose formulae on the left of $\Longrightarrow$.

apply (erule $<$elim-rule$>$)

Elim rule $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ means

➜ If I know A_1 and want to prove A it suffices to show $A_2 \ldots A_n$

Elim rules

Elim rules decompose formulae on the left of $\Longrightarrow$.

apply (erule <elim-rule>)

Elim rule $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ means

- ➜ If I know A_1 and want to prove A it suffices to show $A_2 \ldots A_n$

Applying rule $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ to subgoal C:
Like **rule** but also

- ➜ unifies first premise of rule with an assumption
- ➜ eliminates that assumption

DEMO

MORE PROOF RULES

Iff, Negation, True and False

$$\frac{}{A = B}\ \text{iffI} \qquad \frac{A = B}{C}\ \text{iffE}$$

$$\frac{A = B}{}\ \text{iffD1} \qquad \frac{A = B}{}\ \text{iffD2}$$

$$\frac{}{\neg A}\ \text{notI} \qquad \frac{\neg A}{P}\ \text{notE}$$

Iff, Negation, True and False

NICTA

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \text{iffI} \qquad \frac{A = B}{C}\ \text{iffE}$$

$$\frac{A = B}{}\ \text{iffD1} \qquad \frac{A = B}{}\ \text{iffD2}$$

$$\frac{}{\neg A}\ \text{notI} \qquad \frac{\neg A}{P}\ \text{notE}$$

Iff, Negation, True and False

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \text{iffI} \qquad \frac{A = B \quad [\![A \longrightarrow B; B \longrightarrow A]\!] \Longrightarrow C}{C}\ \text{iffE}$$

$$\frac{A = B}{}\ \text{iffD1} \qquad \frac{A = B}{}\ \text{iffD2}$$

$$\frac{}{\neg A}\ \text{notI} \qquad \frac{\neg A}{P}\ \text{notE}$$

Iff, Negation, True and False

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B}\ \text{iffI} \qquad \frac{A = B \quad [\![A \longrightarrow B; B \longrightarrow A]\!] \Longrightarrow C}{C}\ \text{iffE}$$

$$\frac{A = B}{A \Longrightarrow B}\ \text{iffD1} \qquad \frac{A = B}{B \Longrightarrow A}\ \text{iffD2}$$

$$\frac{}{\neg A}\ \text{notI} \qquad \frac{\neg A}{P}\ \text{notE}$$

Iff, Negation, True and False

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B} \text{ iffI} \qquad \frac{A = B \quad [\![A \longrightarrow B; B \longrightarrow A]\!] \Longrightarrow C}{C} \text{ iffE}$$

$$\frac{A = B}{A \Longrightarrow B} \text{ iffD1} \qquad \frac{A = B}{B \Longrightarrow A} \text{ iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A} \text{ notI} \qquad \frac{\neg A}{P} \text{ notE}$$

Iff, Negation, True and False

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B} \text{ iffI} \qquad \frac{A = B \quad [\![A \longrightarrow B; B \longrightarrow A]\!] \Longrightarrow C}{C} \text{ iffE}$$

$$\frac{A = B}{A \Longrightarrow B} \text{ iffD1} \qquad \frac{A = B}{B \Longrightarrow A} \text{ iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A} \text{ notI} \qquad \frac{\neg A \quad A}{P} \text{ notE}$$

Iff, Negation, True and False

$$\frac{A \Longrightarrow B \quad B \Longrightarrow A}{A = B} \text{ iffI} \qquad \frac{A = B \quad [\![A \longrightarrow B; B \longrightarrow A]\!] \Longrightarrow C}{C} \text{ iffE}$$

$$\frac{A = B}{A \Longrightarrow B} \text{ iffD1} \qquad \frac{A = B}{B \Longrightarrow A} \text{ iffD2}$$

$$\frac{A \Longrightarrow False}{\neg A} \text{ notI} \qquad \frac{\neg A \quad A}{P} \text{ notE}$$

$$\frac{}{\mathit{True}} \text{ TrueI} \qquad \frac{\mathit{False}}{P} \text{ FalseE}$$

Equality

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t}{t = s}\ \text{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \text{trans}$$

Equality

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t}{t = s}\ \text{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \text{trans}$$

$$\frac{s = t \quad P\ s}{P\ t}\ \text{subst}$$

Equality

$$\frac{}{t = t}\ \text{refl} \qquad \frac{s = t}{t = s}\ \text{sym} \qquad \frac{r = s \quad s = t}{r = t}\ \text{trans}$$

$$\frac{s = t \quad P\ s}{P\ t}\ \text{subst}$$

Rarely needed explicitly — used implicitly by term rewriting

Classical

NICTA

$$\frac{}{P = True \lor P = False}\ \text{True-False}$$

Classical

$$\frac{}{P = True \lor P = False}\ \text{True-False}$$

$$\frac{}{P \lor \neg P}\ \text{excluded-middle}$$

$$\frac{\neg A \Longrightarrow False}{A}\ \text{ccontr} \qquad \frac{\neg A \Longrightarrow A}{A}\ \text{classical}$$

Classical

$$\frac{}{P = True \vee P = False} \text{ True-False}$$

$$\frac{}{P \vee \neg P} \text{ excluded-middle}$$

$$\frac{\neg A \Longrightarrow False}{A} \text{ ccontr} \qquad \frac{\neg A \Longrightarrow A}{A} \text{ classical}$$

➜ **excluded-middle**, **ccontr** and **classical** not derivable from the other rules.

Classical

$$\frac{}{P = True \lor P = False} \text{ True-False}$$

$$\frac{}{P \lor \neg P} \text{ excluded-middle}$$

$$\frac{\neg A \Longrightarrow False}{A} \text{ ccontr} \qquad \frac{\neg A \Longrightarrow A}{A} \text{ classical}$$

➜ **excluded-middle**, **ccontr** and **classical** not derivable from the other rules.

➜ if we include True-False, they are derivable

They make the logic “classical”, “non-constructive”

Cases

$$\frac{}{P \vee \neg P} \text{ excluded-middle}$$

is a case distinction on type $bool$

Cases

$$\frac{}{P \vee \neg P}\ \text{excluded-middle}$$

is a case distinction on type $bool$

Isabelle can do case distinctions on arbitrary terms:

apply (case_tac $term$)

Safe and not so safe

Safe rules preserve provability

Safe and not so safe

Safe rules preserve provability

conjI, impI, notI, iffI, refl, ccontr, classical, conjE, disjE

$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

Safe and not so safe

Safe rules preserve provability

conjI, impI, notI, iffI, refl, ccontr, classical, conjE, disjE

$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

Unsafe rules can turn a provable goal into an unprovable one

Safe and not so safe

Safe rules preserve provability

conjI, impI, notI, iffi, refl, ccontr, classical, conjE, disjE

$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

Unsafe rules can turn a provable goal into an unprovable one

disjI1, disjI2, impE, iffD1, iffD2, notE

$$\frac{A}{A \vee B} \text{ disjI1}$$

Safe and not so safe

Safe rules preserve provability

conjI, impI, notI, iffi, refl, ccontr, classical, conjE, disjE

$$\frac{A \quad B}{A \wedge B} \text{ conjI}$$

Unsafe rules can turn a provable goal into an unprovable one

disjI1, disjI2, impE, iffD1, iffD2, notE

$$\frac{A}{A \vee B} \text{ disjI1}$$

Apply safe rules before unsafe ones