Technical Report

UCAM-CL-TR-113
ISSN 1476-2986

Number 113

The representation of logics in higher-order logic

Lawrence C. Paulson

August 1987

ISSN 1476-2986

Contents

1 Introduction

The theorem prover *Isabelle* is based on the idea of composing inference rules to yield derived rules [17]. (See also de Groote [8].) Isabelle regards an inference rule as a *proposition* about the premises and conclusion, not as a function from premises to conclusion (like in LCF). Isabelle supports backwards proof — working from goals to subgoals — representing each state by a rule whose premises are the current subgoals and whose conclusion is the original goal.

The concepts are general enough to allow this one program to handle many logics. Isabelle has been applied to Martin-Löf's Constructive Type Theory, intuitionistic first-order logic, and a classical sequent calculus with ZF set theory. Most of its power comes from derived rules. A rule may require 100 steps to derive but can be applied as a single inference. Reasoning about a defined constant like $\cap$ does not appeal to its complicated definition but to derived rules like

$$\frac{a \in A \qquad a \in B}{a \in A \cap B}$$

Isabelle is under constant development. A stable version, Isabelle-86, implements a naive calculus of proof trees. It is good at proving a desired theorem, not so good at deriving a desired rule. A rule is derived by starting with a conclusion and attempting to reduce it to exactly the desired premises. It would be easier to prove the conclusion taking the premises as assumptions.

Furthermore, it should be possible to assume a rule. In Martin-Löf Type Theory, for example, it makes no sense to prove a theorem about the product type $\Pi_{x\in A} B(x)$ without assuming the axiom A **type** and the rule

$$\frac{a \in A}{B(a)\ \mathbf{type}}$$

Higher-order logic (HOL) is a precise, well-understood formalism that can represent the inference methods of Isabelle-86 and extend them with hypothetical rules. A logic is represented by translating each of its rules into higher-order logic. A few familiar logical connectives suffice: implication handles entailment and assumptions; quantification handles schematic rules and general premises; equality handles definitional equality. In Martin-Löf's terminology, implication forms a hypothetical judgement while quantification forms a general judgement [14].

2 Intuitionistic higher-order logic

Higher-order logic (HOL, also known as simple type theory) is described by Andrews [1]. Several theorem provers use it to advantage [2,7]. HOL is based on the typed λ-calculus [10]. Here is a brief sketch of the fragment required to represent other logics.

2.1 Syntax

Types are sometimes called *arities*, following Martin-Löf, to avoid confusion with ML types or object-level types. Let the Greek letters σ, τ, and υ denote types. The types consist of basic types and function types of the form $\sigma \to \tau$.

The *expressions* are those of the typed λ-calculus — constants, variables, abstractions, combinations — with the usual type constraints. Let a, b, and c denote expressions, and x, y, and z denote variables.

The basic types and constants depend on the logic being represented. But they always include the type of propositions, *prop*, and the logical constants of HOL. A *proposition* is an expression of type *prop*. Let the script letters $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ denote propositions. The logical connectives are proposition-valued functions. The implication $\mathcal{A} \Longrightarrow \mathcal{B}$ means '$\mathcal{A}$ implies $\mathcal{B}$'. The universally quantified proposition $\bigwedge x.\mathcal{A}$ means 'for all x, A is true'. The equality of $a \equiv b$ means 'a equals b'.

The symbols $\Longrightarrow$, $\bigwedge$, and $\equiv$ have been chosen to differ from symbols of *object-logics*: those to be represented in HOL. In predicate logic, implication would be $\supset$, the universal quantifier would be $\forall$. As $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ denote propositions, A, B, and C would denote formulae. But $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are meta-variables denoting propositions of HOL, while A, B, and C are object-variables.

The universal quantifier is represented by λ-abstraction. There is a constant $\bigwedge_\sigma$ of type $(\sigma \to prop) \to prop$ for every type σ. The proposition $\bigwedge x.\mathcal{A}$, where x has type σ, abbreviates $\bigwedge_\sigma(\lambda x.\mathcal{A})$. In particular, $\bigwedge x.\mathcal{A}(x)$ is equivalent to $\bigwedge_\sigma(\mathcal{A})$ by η-conversion.

Abstraction also handles quantifiers in first-order logic, as we shall see in Section 5. The formula $\exists x.A$ is represented by $\exists(\lambda x.A)$, where $\exists$ is a constant symbol of type $(term \to form) \to form$.

2.2 Syntactic conventions

The application of a to the successive arguments $b_1, \ldots, b_m$ is written $a(b_1, \ldots, b_m)$:

$$a(b_1, \ldots, b_m) \quad \text{abbreviates} \quad (\cdots(ab_1)\cdots b_m)$$

Implication associates to the right:

$$\mathcal{A}_1 \Longrightarrow \cdots \Longrightarrow \mathcal{A}_m \Longrightarrow \mathcal{B} \quad \text{abbreviates} \quad \mathcal{A}_1 \Longrightarrow (\cdots \Longrightarrow (\mathcal{A}_m \Longrightarrow \mathcal{B})\cdots)$$

Let the script letters $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ denote lists of propositions. If $\mathcal{G}$ is the list $[\mathcal{A}_1, \ldots, \mathcal{A}_m]$, then

$$\mathcal{G} \Longrightarrow \mathcal{B} \quad \text{abbreviates} \quad \mathcal{A}_1 \Longrightarrow (\cdots \Longrightarrow (\mathcal{A}_m \Longrightarrow \mathcal{B})\cdots)$$

and also

$$[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{B} \quad \text{abbreviates} \quad \mathcal{A}_1 \Longrightarrow (\cdots \Longrightarrow (\mathcal{A}_m \Longrightarrow \mathcal{B})\cdots)$$

One quantifier does the work of many:

$$\bigwedge x_1 \ldots x_m . \mathcal{A} \quad \text{abbreviates} \quad \bigwedge x_1 . \ldots \bigwedge x_m . \mathcal{A}$$

The scope of a quantifier extends as far as possible to the right:

$$\bigwedge x . \mathcal{A} \Longrightarrow b \equiv c \quad \text{abbreviates} \quad \bigwedge x . (\mathcal{A} \Longrightarrow (b \equiv c))$$

A *substitution* has the form $[a_1/x_1, \ldots, a_k/x_k]$, where $x_1, \ldots, x_k$ are distinct variables, and $a_1, \ldots, a_k$ are expressions. If b is an expression and θ is the substitution above then $b\theta$ is the expression that results from simultaneously replacing every free occurence of x_i by a_i in b, for $i = 1, \ldots, k$, renaming bound variables to avoid capture.

Note that the combination $a(b)$ denotes the application of a to b, not substitution. The law of β-reduction, namely $(\lambda x.a)(b) \equiv a[b/x]$, handles substitution at the object-level. Read $a[b/x]$ as 'a putting b for x.' Substitutions are used in the discussion; they are not part of HOL itself.

2.3 Semantics

Higher-order logic was developed to formalize the foundations of mathematics. Its consistency is proved by interpreting it in set theory, where every type denotes a non-empty set. The interpretations of the basic types must be given; that of $\sigma \rightarrow \tau$ is the set of functions from σ to τ. The type *prop* denotes a set of truth values: classical logic would use $\{T, F\}$; an intuitionistic interpretation is also possible.

A closed expression of type σ denotes a value of the corresponding set. The logical constants $\bigwedge_\sigma$, $\Longrightarrow$, and $\equiv_\sigma$ have standard meanings.

2.4 Inference rules

The constant symbols include, for every type σ,

$$\begin{aligned} \Longrightarrow &: \mathit{prop} \rightarrow \mathit{prop} \rightarrow \mathit{prop} \\ \bigwedge_\sigma &: (\sigma \rightarrow \mathit{prop}) \rightarrow \mathit{prop} \\ \equiv_\sigma &: \sigma \rightarrow \sigma \rightarrow \mathit{prop} \end{aligned}$$

Higher-order logic is usually formalized in Hilbert style. Natural deduction rules are more convenient. These are derivable; for example, $\Longrightarrow$-introduction corresponds to the deduction theorem [1].

Implication:

$$\frac{\begin{matrix}[\mathcal{A}] \\ \mathcal{B}\end{matrix}}{\mathcal{A} \Longrightarrow \mathcal{B}} \qquad (\mathit{introduction})$$

$$\frac{\mathcal{A} \Longrightarrow \mathcal{B} \qquad \mathcal{A}}{\mathcal{B}} \qquad (\mathit{elimination})$$

Universal quantification:

$$\frac{\mathcal{A}[y/x]}{\bigwedge x.\mathcal{A}} \qquad (\textit{introduction})$$

$$\frac{\bigwedge x.\mathcal{A}}{\mathcal{A}[b/x]} \qquad (\textit{elimination})$$

These are also called *generalization* and *specialization.* The generalization rule is subject to the proviso that y is not free in the assumptions or $\mathcal{A}$.

Equality:

$$a \equiv a \qquad (\textit{reflexivity})$$

$$\frac{a \equiv b}{b \equiv a} \qquad (\textit{symmetry})$$

$$\frac{a \equiv b \qquad b \equiv c}{a \equiv c} \qquad (\textit{transitivity})$$

$$\lambda x.a \equiv \lambda y.a[y/x] \qquad (\alpha\textit{-conversion})$$

The α-conversion axiom holds provided y is not free in a.

$$(\lambda x.a)(b) \equiv a[b/x] \qquad (\beta\textit{-conversion})$$

$$\frac{a(y) \equiv b(y)}{a \equiv b} \qquad (\textit{extensionality})$$

$$\frac{a[y/x] \equiv b[y/x]}{\lambda x.a \equiv \lambda x.b} \qquad (\textit{abstraction})$$

$$\frac{a \equiv b \qquad c \equiv d}{a(c) \equiv b(d)} \qquad (\textit{combination})$$

The extensionality and abstraction rules hold provided y is not free in the assumptions, a, or b.

A rule giving logical equivalence from equality is

$$\frac{\mathcal{A} \equiv \mathcal{B} \qquad \mathcal{A}}{\mathcal{B}}$$

Every expression has a unique normal form, where $a \equiv b$ if and only if a and b have the same normal form [10].

3 Representing intuitionistic propositional logic

To represent the syntax of intuitionistic propositional logic (IPL), introduce the basic type *form* of formulae, the object-variables $A, B, C : \mathit{form}$, and the constant symbols

$$\begin{aligned} \bot &: \mathit{form} \\ \wedge, \vee, \supset &: \mathit{form} \to (\mathit{form} \to \mathit{form}) \\ \mathrm{true} &: \mathit{form} \to \mathit{prop} \end{aligned}$$

Object-sentences are enclosed in double brackets $[\![\,]\!]$ for emphasis. The proposition $[\![A]\!]$ denotes true(A) and means that A is true. The constant 'true' converts a formula to a proposition. Letting $[\![A]\!]$ denote A would require the types *form* and *prop* to be the same, confusing formulae of the object-logic with propositions of the meta-logic.

The natural deduction rules of intuitionistic logic correspond to the following axioms, called the IPL axioms. Outer universal quantifiers are omitted. For example, the axiom for $\wedge$-introduction is really

$$\bigwedge AB \,.\, [\![A]\!] \Longrightarrow [\![B]\!] \Longrightarrow [\![A \wedge B]\!]$$

Conjunction:

$$[\![A]\!] \Longrightarrow [\![B]\!] \Longrightarrow [\![A \wedge B]\!] \qquad (\textit{introduction})$$

$$[\![A \wedge B]\!] \Longrightarrow [\![A]\!] \qquad [\![A \wedge B]\!] \Longrightarrow [\![B]\!] \qquad (\textit{elimination})$$

Disjunction:

$$[\![A]\!] \Longrightarrow [\![A \vee B]\!] \qquad [\![B]\!] \Longrightarrow [\![A \vee B]\!] \qquad (\textit{introduction})$$

$$[\![A \vee B]\!] \Longrightarrow ([\![A]\!] \Longrightarrow [\![C]\!]) \Longrightarrow ([\![B]\!] \Longrightarrow [\![C]\!]) \Longrightarrow [\![C]\!] \qquad (\textit{elimination})$$

Implication:

$$([\![A]\!] \Longrightarrow [\![B]\!]) \Longrightarrow [\![A \supset B]\!] \qquad (\textit{introduction})$$

$$[\![A \supset B]\!] \Longrightarrow [\![A]\!] \Longrightarrow [\![B]\!] \qquad (\textit{elimination})$$

Contradiction:

$$[\![\bot]\!] \Longrightarrow [\![A]\!]$$

To interpret the new symbols, let the type *form* denote a set of truth values such that the logical constants $\wedge, \vee, \supset, \bot$ have their intuitionistic meanings. The axioms are true under this interpretation: for example, if A is true and B is true then $A \wedge B$ is true. Each axiom formalizes the justification of the corresponding rule. This observation is obvious, but fundamental; I am grateful to Martin Hyland for it.

Meta-level implication, $\Longrightarrow$, expresses the discharge of assumptions. The $\supset$-introduction axiom says that if the truth of A implies the truth of B, then the formula $A \supset B$ is true. An obvious question is whether this is a faithful representation of the object-logic in HOL.

Definition 1 A set of axioms $\mathcal{G}$ *faithfully expresses* an object-logic when the following holds: there is an object-proof of B from $A_1, \ldots, A_m$ if and only if there is an HOL-proof of $[\![B]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$ and $\mathcal{G}$.

Particularly important is soundness. If $[\![A_1]\!], \ldots, [\![A_m]\!]$ imply $[\![B]\!]$ in HOL, then $A_1, \ldots, A_m$ imply B in IPL because the axioms are true of IPL, and the rules of higher-order logic are sound. A better proof is by induction on normal proof trees in HOL. A normal proof tree consists entirely of the application of axioms to elimination rules (representing application of the corresponding object-rules) followed by introduction rules (representing the discharge of assumptions). Here is a summary of the concepts as described by Prawitz [19,20]. For simplicity, let us ignore equality rules, identifying expressions that are equivalent up to λ-conversions.

Every proof tree in higher-order logic can be *normalized*: modified such that no elimination rule immediately follows the corresponding introduction rule. A *branch* in a proof tree is obtained by repeatedly walking downwards from the first premise of a rule to its conclusion; this process terminates when the conclusion is not the first premise of another rule. In a normal proof, every branch begins with an assumption or axiom, then has a series of eliminations, then a series of introductions. During the eliminations the formulae become smaller and smaller until they reach a minimum; during the introductions they become larger again.

Observe that $[\![B]\!]$ is an atomic HOL formula. A normal proof tree can be put into *expanded* normal form, where every minimum formula is atomic [20, page 254]. For example, if a minimum formula is $\mathcal{A} \Longrightarrow \mathcal{B}$, then the following can be spliced into the proof, reducing the minimum formula to $\mathcal{B}$:

$$\dfrac{\dfrac{\mathcal{A} \Longrightarrow \mathcal{B} \qquad [\mathcal{A}]}{\mathcal{B}}}{\mathcal{A} \Longrightarrow \mathcal{B}}$$

Theorem 1 (Soundness) *If there is an* HOL *proof of* $[\![B]\!]$ *from* $[\![A_1]\!], \ldots, [\![A_m]\!]$ *and the* IPL *axioms, then there is an* IPL *proof of* B *from* $A_1, \ldots, A_m$.
Proof: By induction on the size of the expanded normal proof tree. Since $[\![B]\!]$ is atomic, the branch terminating with $[\![B]\!]$ cannot contain introduction rules, and cannot discharge assumptions. The branch must consist entirely of elimination rules. If it is just $[\![B]\!]$, then B is one of $A_1, \ldots, A_m$, and is trivially provable. Otherwise the branch contains elimination rules, so its first formula cannot be atomic. It must consist of an axiom, followed by elimination rules, reducing it to B. There is one case for each axiom.

For the $\wedge$-introduction axiom, B is $C \wedge D$ for some formulae C and D. The proof begins with two $\bigwedge$-eliminations, replacing the universal variables by C and D. Then two $\Longrightarrow$-eliminations, applied to proofs of $[\![C]\!]$ and $[\![D]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$, prove $[\![C \wedge D]\!]$. By the induction hypothesis, there are IPL proofs of C and D from $A_1, \ldots, A_m$. Applying $\wedge$-introduction gives an IPL proof of $C \wedge D$.

For $\supset$-introduction, B is $C \supset D$. The proof begins with $\bigwedge$-eliminations for the formulae C and D. Then $\Longrightarrow$-elimination, applied to a proof of $[\![C]\!] \Longrightarrow [\![D]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$, proves $[\![C \supset D]\!]$. Since the tree is in expanded normal form, the proof of $[\![C]\!] \Longrightarrow [\![D]\!]$ must consist of a proof of $[\![D]\!]$ followed by $\Longrightarrow$-introduction, discharging the assumption $[\![C]\!]$:

$$\frac{\begin{array}{c}[\,[\![C]\!]\,]\\ \vdots \\ [\![D]\!]\end{array}}{[\![C]\!] \Longrightarrow [\![D]\!]}$$

By the induction hypothesis, there is an IPL proof of D from $A_1, \ldots, A_m, C$, and $\supset$-introduction gives an IPL proof of $C \supset D$ from $A_1, \ldots, A_m$.

The cases for the other axioms are similar. □

Theorem 2 (Completeness) *If there is an* IPL *proof of B from $A_1, \ldots, A_m$, then there is an* HOL *proof of $[\![B]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$ and the* IPL *axioms.*
Proof: By induction on the size of the proof tree with root B and assumptions $A_1, \ldots, A_m$, we can construct a proof of $[\![B]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$ in HOL.

Suppose the last inference of the IPL proof is $\supset$-introduction, and the conclusion is $C \supset D$. Then the rule is applied to an IPL proof of D from C. By the induction hypothesis, there is an HOL proof of $[\![D]\!]$ from $[\![C]\!]$, and $\Longrightarrow$-introduction concludes $[\![C]\!] \Longrightarrow [\![D]\!]$. Then axiom for $\supset$-introduction proves $[\![C \supset D]\!]$, discharging the assumption $[\![C]\!]$.

The cases for the other axioms are similar. □

To summarize:

Theorem 3 (Faithfulness) *The* IPL *axioms faithfully express* IPL.

4 The development of backwards proofs

It is often natural to construct a proof backwards. The reduction of the goal $\mathcal{A}$ to the subgoals $\mathcal{A}_1, \ldots, \mathcal{A}_m$ corresponds to the derived rule $\mathcal{A}_1, \ldots, \mathcal{A}_m/\mathcal{A}$, Higher-order logic represents this as the implication $\mathcal{A}_1 \Longrightarrow \cdots \Longrightarrow \mathcal{A}_m \Longrightarrow \mathcal{A}$. A resolution rule for HOL combines such implications. This implemention of backwards proof is unusual, but has unique advantages.

$$\frac{A \land B \supset C \supset A \land C}{A \land B \supset C \supset A \land C} \overset{\to\mathrm{I}}{\longmapsto} \frac{\begin{matrix} A \land B \\ \vdots \\ C \supset A \land C \end{matrix}}{A \land B \supset C \supset A \land C} \overset{\to\mathrm{I}}{\longmapsto} \frac{\begin{matrix} A \land B \\ C \\ \vdots \\ A \land C \end{matrix}}{A \land B \supset C \supset A \land C}$$

$$\overset{\land\mathrm{I}}{\longmapsto} \frac{\begin{matrix} A \land B \\ C \\ \vdots \\ A \end{matrix} \qquad \begin{matrix} A \land B \\ C \\ \vdots \\ C \end{matrix}}{A \land B \supset C \supset A \land C} \overset{\mathrm{asm}}{\longmapsto} \frac{\begin{matrix} A \land B \\ C \\ \vdots \\ A \end{matrix}}{A \land B \supset C \supset A \land C} \overset{\land\mathrm{E}}{\longmapsto} \frac{\begin{matrix} A \land B \\ C \\ A \\ \vdots \\ A \end{matrix}}{A \land B \supset C \supset A \land C}$$

$$\overset{\mathrm{asm}}{\longmapsto} \quad A \land B \supset C \supset A \land C$$

Figure 1: The steps of the construction of the proof tree

4.1 Partial proofs as derived rules

The method is illustrated by an example: a proof of $A \land B \supset (C \supset A \land C)$. The proof tree:

$$\cfrac{\cfrac{\cfrac{\cfrac{[A \land B]}{A} \qquad [C]}{A \land C}}{C \supset A \land C}}{A \land B \supset (C \supset A \land C)}$$

A backwards proof is found by working upwards. Every partial proof is represented by a derived rule whose conclusion is the ultimate goal, here $A \land B \supset (C \supset A \land C)$, and whose premises are the current subgoals; its internal structure, which has no further role to play, is suppressed. The initial partial proof is represented by the trivial rule whose premise and conclusion are the ultimate goal.

Figure 1 shows the sequence of partial proofs. The initial partial proof is combined with $\supset$-introduction, which gives rise to the assumption $A \land B$. A second $\supset$-introduction adds C to the assumptions. Then $\land$-introduction splits in two the goal, namely $A \land C$. The full proof tree at this point is

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}A \land B & & A \land B\\ C & & C\\ \vdots & & \vdots\\ A & & C\end{matrix}}{A \land C}}{C \supset A \land C}}{A \land B \supset (C \supset A \land C)}}{A \land B \supset (C \supset A \land C)}$$

The second subgoal, C, holds trivially by assumption. Then $\land$-elimination splits the assumption $A \land B$, solving the first subgoal. The derivation ends with the theorem $A \land B \supset (C \supset A \land C)$.

Isabelle-86 represents backwards proof similarly but does not handle assumptions for natural deduction. If the object-logic allows the discharge of assumptions, it must express this through a sequent formulation.

4.2 A formalization of partial proofs

This proof tree construction can be formalized in higher-order logic. Horizontal lines, used in Figure 1 to indicate object-level inferences, now indicate meta-level inferences; object-level inferences are expressed using $\Longrightarrow$. The initial proof state is the trivial theorem $\mathcal{C} \Longrightarrow \mathcal{C}$; a state of the proof with n subgoals is represented by the theorem

$$[\mathcal{B}_1, \ldots, \mathcal{B}_n] \Longrightarrow \mathcal{C}$$

The proof state with zero subgoals is the theorem $\mathcal{C}$.

A partial proof is a theorem. It holds even if the goal or some subgoals are false. Devising a representation of partial proofs that works regardless of the truth of the goals has been difficult — particularly for quantifiers, as we shall see.

Each proof step uses a resolution rule, like in Isabelle-86. In the simplest form of resolution, the substitution θ must match $\mathcal{A}$ against $\mathcal{B}$, namely $\mathcal{A}\theta \equiv \mathcal{B}$. The conclusion is put into normal form:

$$\frac{[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A} \qquad \mathcal{F} \Longrightarrow \mathcal{B} \Longrightarrow \mathcal{C}}{\mathcal{F} \Longrightarrow [\mathcal{A}_1\theta, \ldots, \mathcal{A}_m\theta] \Longrightarrow \mathcal{C}} \tag{1}$$

The notation avoids some subscripts by letting $\mathcal{F}$ stand for a list of propositions. Here is the same rule again, writing out the lists in full. If $\mathcal{A}\theta \equiv \mathcal{B}_i$ then

$$\frac{[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A} \qquad [\mathcal{B}_1, \ldots, \mathcal{B}_n] \Longrightarrow \mathcal{D}}{[\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}, \mathcal{A}_1\theta, \ldots, \mathcal{A}_m\theta, \mathcal{B}_{i+1}, \ldots, \mathcal{B}_n] \Longrightarrow \mathcal{D}}$$

This shows more clearly how the rule acts on a proof state: the subgoal $\mathcal{B}_i$ is replaced by $\mathcal{A}_1\theta, \ldots, \mathcal{A}_m\theta$.

The first premise is an object-level rule and the second is a proof state; the conclusion is a new proof state:

$$\frac{\text{object-level rule} \qquad \text{proof state}}{\text{new proof state}}$$

The general form of resolution will involve the unification of $\mathcal{A}$ with $\mathcal{B}_i$, so the free variables of the object-level rule should be distinct from those of the proof state. In the examples, variables will be renamed by subscripting.

Resolution is easily derived from the rules of higher-order logic. Using both quantifier rules k times derives an *instantiation* rule,

$$\frac{\mathcal{A}}{\mathcal{A}[a_1/x_1, \ldots, a_k/x_k]}$$

provided that $x_1, \ldots, x_k$ are not free in the assumptions. Resolution consists of the instantiation of its premises followed by reasoning about implication.

4.3 Formalizing the use of assumptions

Let us return to the sample proof of $A \wedge B \supset (C \supset A \wedge C)$. The first step in the formalized version is the resolution

$$\frac{([\![A_1]\!] \Longrightarrow [\![B_1]\!]) \Longrightarrow [\![A_1 \supset B_1]\!] \qquad [\![A \wedge B \supset (C \supset A \wedge C)]\!] \Longrightarrow [\![A \wedge B \supset (C \supset A \wedge C)]\!]}{([\![A \wedge B]\!] \Longrightarrow [\![C \supset A \wedge C]\!]) \Longrightarrow [\![A \wedge B \supset (C \supset A \wedge C)]\!]}$$

The $\supset$-introduction axiom is resolved with the initial proof state, instantiating the variable A_1 to $A \wedge B$ and B_1 to $C \supset A \wedge C$. The new state has one subgoal: to prove $C \supset A \wedge C$ from the assumption $A \wedge B$.

The next step requires a new rule to handle assumptions. The resolution rule expects a theorem of the form $\cdots \Longrightarrow ([\![A \wedge B]\!] \Longrightarrow [\![C \supset A \wedge C]\!])$ as its first premise. One way to obtain this is by 'lifting' an object-level rule, represented by the implication $[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A}$, over a list of assumptions $\mathcal{H}$:

$$\frac{[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A}}{[\mathcal{H} \Longrightarrow \mathcal{A}_1, \ldots, \mathcal{H} \Longrightarrow \mathcal{A}_m] \Longrightarrow (\mathcal{H} \Longrightarrow \mathcal{A})}$$

The lifting rule is derived by the implication rules of higher-order logic.

Combining the lifting rule with the earlier resolution rule (1) gives a form of resolution that handles assumptions in subgoals:

$$\frac{[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A} \qquad \mathcal{F} \Longrightarrow (\mathcal{H} \Longrightarrow \mathcal{B}) \Longrightarrow \mathcal{C}}{\mathcal{F} \Longrightarrow [\mathcal{H} \Longrightarrow \mathcal{A}_1\theta, \ldots, \mathcal{H} \Longrightarrow \mathcal{A}_m\theta] \Longrightarrow \mathcal{C}} \tag{2}$$

The rule holds provided $\mathcal{A}\theta \equiv \mathcal{B}$. It replaces the subgoal $\mathcal{H} \Longrightarrow \mathcal{B}$ by $\mathcal{H} \Longrightarrow \mathcal{A}_1\theta, \ldots, \mathcal{H} \Longrightarrow \mathcal{A}_m\theta$: the assumptions of $\mathcal{B}$ are passed to the new subgoals.

For an example of this new rule, let us resolve the $\supset$-introduction axiom with the current proof state, namely

$$([\![A \wedge B]\!] \Longrightarrow [\![C \supset A \wedge C]\!]) \Longrightarrow [\![A \wedge B \supset (C \supset A \wedge C)]\!]$$

Lifting the axiom over the assumption $A \land B$ yields

$$\frac{(\llbracket A_2 \rrbracket \Longrightarrow \llbracket B_2 \rrbracket) \Longrightarrow \llbracket A_2 \supset B_2 \rrbracket}{(A \land B \Longrightarrow \llbracket A_2 \rrbracket \Longrightarrow \llbracket B_2 \rrbracket) \Longrightarrow (A \land B \Longrightarrow \llbracket A_2 \supset B_2 \rrbracket)}$$

Resolving the conclusion with the proof state instantiates A_2 to C and B_2 to $A \land C$. The new state is

$$(\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \land C \rrbracket) \Longrightarrow \llbracket A \land B \supset (C \supset A \land C) \rrbracket$$

The next step is resolution with the axiom of $\land$-introduction, lifting over the assumptions $A \land B$ and C. Since the proof state takes up too much room, let us omit it when necessary: 'current proof state' refers to the conclusion of the previous resolution.

$$\frac{\llbracket A_3 \rrbracket \Longrightarrow \llbracket B_3 \rrbracket \Longrightarrow \llbracket A_3 \land B_3 \rrbracket \qquad \text{current proof state}}{(\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow (\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket) \Longrightarrow \llbracket A \land B \supset (C \supset A \land C) \rrbracket}$$

The variable instantiations are A_3 to A and B_3 to C. Observe how the assumptions are copied to both subgoals.

The next step, using assumption C in the second goal, takes place by resolution with the HOL theorem $\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket$.

$$\frac{\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket \qquad \text{current proof state}}{(\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow \llbracket A \land B \supset (C \supset A \land C) \rrbracket}$$

Next comes resolution with $\land$-elimination. We have no rule that can use the assumption $A \land B$ to produce a new assumption A, so we must deviate from the earlier proof. The goal A is instead reduced to the subgoal $A \land B$. Since the goal does not determine B, the proof uses a particular instance of the $\land$-elimination axiom, lifted over $A \land B$ and C:

$$\frac{\llbracket A \land B \rrbracket \Longrightarrow \llbracket A \rrbracket \qquad \text{current proof state}}{(\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \land B \rrbracket) \Longrightarrow \llbracket A \land B \supset (C \supset A \land C) \rrbracket}$$

Unification can determine the necessary instance of B. But a better treatment of $\land$-elimination is given in Section 4.5.

The next step uses the assumption $A \land B$ in the goal, by resolution with the theorem $\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \land B \rrbracket$.

$$\frac{\llbracket A \land B \rrbracket \Longrightarrow \llbracket C \rrbracket \Longrightarrow \llbracket A \land B \rrbracket \qquad \text{current proof state}}{\llbracket A \land B \supset (C \supset A \land C) \rrbracket}$$

This concludes the proof of $A \land B \supset (C \supset A \land C)$, representing each step of the object-level proof by a use of resolution. It is not hard to see that every proof in the current object-logic can be represented by a resolution proof like the one above.

4.4 Deriving rules in an object-logic

To derive a rule, its premises are taken as assumptions, and finally discharged via $\Longrightarrow$-introduction. Our example of proving an implicative theorem is easily changed to that of deriving the rule

$$\frac{A \wedge B}{C \supset A \wedge C}$$

The first step is resolution with the axiom of $\supset$-introduction., instantiating A_1 to C and B_1 to $A \wedge C$:

$$\frac{(\llbracket A_1 \rrbracket \Longrightarrow \llbracket B_1 \rrbracket) \Longrightarrow \llbracket A_1 \supset B_1 \rrbracket \qquad \llbracket C \supset A \wedge C \rrbracket \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}{(\llbracket C \rrbracket \Longrightarrow \llbracket A \wedge C \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}$$

Next comes resolution with $\wedge$-introduction, lifting over the assumption C:

$$\frac{\llbracket A_2 \rrbracket \Longrightarrow \llbracket B_2 \rrbracket \Longrightarrow \llbracket A_2 \wedge B_2 \rrbracket \qquad (\llbracket C \rrbracket \Longrightarrow \llbracket A \wedge C \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}{(\llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow (\llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}$$

The next resolution uses the assumption C to solve the second goal:

$$\frac{\llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket \qquad (\llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow (\llbracket C \rrbracket \Longrightarrow \llbracket C \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}{(\llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}$$

The next resolution uses $\wedge$-elimination, lifting over C:

$$\frac{\llbracket A \wedge B \rrbracket \Longrightarrow \llbracket A \rrbracket \qquad (\llbracket C \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}{(\llbracket C \rrbracket \Longrightarrow \llbracket A \wedge B \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}$$

Here we diverge from the previous proof. The subgoal does not mention the assumption $A \wedge B$. Instead, this assumption is made at the meta-level. We resolve with $A \wedge B$, lifting over C.

$$\frac{[\,\llbracket A \wedge B \rrbracket\,] \qquad (\llbracket C \rrbracket \Longrightarrow \llbracket A \wedge B \rrbracket) \Longrightarrow \llbracket C \supset A \wedge C \rrbracket}{\llbracket C \supset A \wedge C \rrbracket}$$

Finally $\Longrightarrow$-introduction discharges the assumption $\llbracket A \wedge B \rrbracket$, yielding

$$\llbracket A \wedge B \rrbracket \Longrightarrow \llbracket C \supset A \wedge C \rrbracket$$

This completes the proof.

It is also possible to assume a rule. Adding the double negation law

$$\frac{(A \supset \bot) \supset \bot}{A}$$

to intuitionistic propositional logic gives classical logic. We can investigate the consequences of adding this rule; it is represented by the proposition

$$\bigwedge A.\, \llbracket (A \supset \bot) \supset \bot \rrbracket \Longrightarrow \llbracket A \rrbracket$$

The double negation law implies the excluded middle, a fact represented by the following theorem:

$$(\bigwedge A.\, \llbracket (A \supset \bot) \supset \bot \rrbracket \Longrightarrow \llbracket A \rrbracket) \Longrightarrow \bigwedge B.\, \llbracket B \vee (B \supset \bot) \rrbracket$$

4.5 Deriving rules for forwards proof

The $\wedge$-elimination rules typically work in the forwards direction, producing a new theorem (or assumption) from an existing one:

$$\frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

In backwards proof, it is strange to reduce the goal A to the subgoal $A \wedge B$.

We could derive a meta-rule to give forwards proof using the assumptions of a subgoal. But there is another version of $\wedge$-elimination better suited to backwards proof. It fits the standard pattern of elimination rules, such as those for disjunction and existential quantifiers [21]. It applies to any goal C, reducing it to the subgoal of proving $A \wedge B$ and the subgoal of proving C assuming A and B:

$$\frac{A \wedge B \qquad \begin{array}{c}[A, B]\\ C\end{array}}{C}$$

In higher-order logic, the rule is

$$[\![A \wedge B]\!] \Longrightarrow ([\![A]\!] \Longrightarrow [\![B]\!] \Longrightarrow [\![C]\!]) \Longrightarrow [\![C]\!]$$

It is easily proved by $\Longrightarrow$-introduction, $\Longrightarrow$-elimination, and the axioms

$$[\![A \wedge B]\!] \Longrightarrow [\![A]\!] \qquad [\![A \wedge B]\!] \Longrightarrow [\![B]\!]$$

It is best proved directly — the techniques of the previous section seem cumbersome — because it is not a proof *in* the object-logic, but *about* the object-logic.

The inference rules of LCF's object-logic are given as functions from theorems to theorems. This provides forwards proof. For backwards proof, an LCF tactic maps a goal to a list of subgoals paired with a validation function, which maps theorems to theorems [18]. An *invalid* tactic is one that promises more than it can deliver. It reduces a goal A to a set of subgoals that do not imply A. The error may not be noticed until the end of the proof, when the validation function fails to produce the expected theorem.

Representing the goal tree by a derived rule in a meta-logic eliminates the danger of invalid tactics. Later we will see another advantage: it allows goals containing variables, instantiated by unification. But Isabelle may need LCF-style tactics for reasoning in higher-order logic, to derive the conjunction rule given above. Isabelle must support both object-level proof and meta-level proof.

5 Representing quantifiers

Adding quantifiers to the previous object-logic gives intuitionistic first-order logic (IFOL). Let us introduce the basic type *term* of terms, variables ranging over terms, and propositional functions. By convention, primes indicate the arity of a symbol: A' is a function taking one argument, A'' takes two, etc. But we can often be causal about this, for in $A(x)$ clearly A must have function type.

Introduce the type *term*, the variables

$$\begin{aligned} t, u, x, y, z &: \mathit{term} \\ A', B', C' &: \mathit{term} \to \mathit{form} \end{aligned}$$

and the constant symbols

$$\forall, \exists \ : \ (\mathit{term} \to \mathit{form}) \to \mathit{form}$$

Let us write $\forall x.A$ for $\forall(\lambda x.A)$ and $\exists x.A$ for $\exists(\lambda x.A)$. The IFOL axioms consist of the IPL axioms, for propositional logic, plus the following axioms, for quantifiers:

Universal quantification:

$$(\bigwedge x.[\![A'(x)]\!]) \Longrightarrow [\![\forall x.A'(x)]\!] \qquad (\mathit{introduction})$$

$$[\![\forall x.A'(x)]\!] \Longrightarrow [\![A'(t)]\!] \qquad (\mathit{elimination})$$

Existential quantification:

$$[\![A'(t)]\!] \Longrightarrow [\![\exists x.A'(x)]\!] \qquad (\mathit{introduction})$$

$$[\![\exists x.A'(x)]\!] \Longrightarrow (\bigwedge x.\, [\![A'(x)]\!] \Longrightarrow [\![B]\!]) \Longrightarrow [\![B]\!] \qquad (\mathit{elimination})$$

To see why these are true requires a semantics of first-order logic. The type *term* denotes a set of individuals; the quantifiers have standard meanings. The axiom for $\forall$-introduction simply asserts one direction of the basic property of the universal quantifer: $A'(x)$ is true for all x if, and only if, $\forall x.A'(x)$ is true. The axiom for $\forall$-elimination asserts the converse. The axiom for $\exists$-elimination states that if $A'(x)$ implies B for every x, then $\exists x.A'(x)$ implies B. This is correct because if $\exists x.A'(x)$ is true then there is an x such that $A'(x)$ is true, and so B is true. Each axiom formalizes the standard justification of the object-rule.

The proof that these axioms faithfully represent first-order logic is similar to that for propositional logic.

Theorem 4 (Soundness) *If there is an* HOL *proof of* $[\![B]\!]$ *from* $[\![A_1]\!], \ldots, [\![A_m]\!]$ *and the* IFOL *axioms, then there is an* IFOL *proof of* B *from* $A_1, \ldots, A_m$.

Proof: By induction on the size of the expanded normal proof tree. The branch terminating with $[\![B]\!]$, if it is non-trivial, consists of an axiom followed by elimination rules.

For the $\exists$-introduction axiom, B is $\exists x.C'(x)$. Two $\bigwedge$-eliminations replace the universal variables by C' and u. Then $\Longrightarrow$-elimination is applied to a proof of $[\![C'(u)]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$. By the induction hypothesis there is an IFOL proof of $C'(u)$ from $A_1, \ldots, A_m$, and $\exists$-introduction proves $\exists x.C'(x)$.

The hardest case is $\exists$-elimination. The proof contains $\Longrightarrow$-eliminations applied to proofs of $[\![\exists x.C'(x)]\!]$ and $\bigwedge x.\, [\![C'(x)]\!] \Longrightarrow [\![D]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$. Since the tree is in expanded normal form, the latter proof consists of a proof of $[\![D]\!]$ followed by $\Longrightarrow$-introduction, discharging the assumption $[\![C'(y)]\!]$ — for some y not free in $[\![A_1]\!], \ldots, [\![A_m]\!]$ — followed by $\bigwedge$-introduction:

$$\dfrac{\dfrac{\begin{array}{c}[\,[\![C'(y)]\!]\,]\\ \vdots \\ [\![D]\!]\end{array}}{[\![C'(y)]\!] \Longrightarrow [\![D]\!]}}{\bigwedge x.\, [\![C'(x)]\!] \Longrightarrow [\![D]\!]}$$

By the induction hypothesis, there are IFOL proofs of D from $A_1, \ldots, A_m, C'(y)$ and of $\exists x.C'(x)$ from $A_1, \ldots, A_m$. The $\exists$-elimination rule gives an IFOL proof of D from $A_1, \ldots, A_m$.

The cases for the other axioms are similar. □

Theorem 5 (Completeness) *If there is an* IFOL *proof of B from $A_1, \ldots, A_m$, then there is an* HOL *proof of $[\![B]\!]$ from $[\![A_1]\!], \ldots, [\![A_m]\!]$ and the* IFOL *axioms.*
Proof: By induction over an IFOL proof tree with root B and assumptions $A_1, \ldots, A_m$. The hardest case is when the last inference is $\exists$-elimination. Then the rule is applied to an IFOL proof of $\exists x.C(x)$, and to a proof of B from $C(y)$, where the variable y is not free in the other assumptions:

$$\dfrac{\begin{array}{c}\vdots \\ \exists x.C(x)\end{array} \qquad \begin{array}{c}C(y)\\ \vdots \\ B\end{array}}{B}$$

By the axiom for $\exists$-elimination, it is enough to prove the theorems $[\![\exists x.C(x)]\!]$ and $\bigwedge x.\, [\![C(x)]\!] \Longrightarrow [\![B]\!]$. By the induction hypothesis, there is an HOL proof of $[\![\exists x.C(x)]\!]$, and also a proof of $[\![B]\!]$ from $[\![C(y)]\!]$. By $\Longrightarrow$- and $\bigwedge$-introduction, since y is not free in the assumptions, we have $\bigwedge y.\, [\![C(y)]\!] \Longrightarrow [\![B]\!]$. □

To summarize:

Theorem 6 (Faithfulness) *The* IFOL *axioms faithfully express* IFOL.

6 Extending resolution to quantifiers

So far we have considered propositional logic. Quantifiers complicate the situation by introducing universal goals and free variables in goals.

6.1 Lifting over universal quantifiers

A universal goal has the form $\bigwedge x_1 \ldots x_n.\mathcal{A}$. It arises from trying to prove a universal premise, like that of $\forall$-introduction or $\exists$-elimination. Consider a proof of $\forall x.A(x) \vee B(x)$ from $\forall x.A(x)$:

$$\cfrac{\cfrac{\cfrac{\forall x.A(x)}{A(x)}}{A(x) \vee B(x)}}{\forall x.A(x) \vee B(x)}$$

Working backwards, the first inference is $\forall$-introduction. The first resolution step of the formalized proof instantiates A_1' to $\lambda x.A(x) \vee B(x)$:

$$\frac{(\bigwedge x.[\![A_1'(x)]\!]) \Longrightarrow [\![\forall x.A_1'(x)]\!] \qquad [\![\forall x.A(x) \vee B(x)]\!] \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]}{(\bigwedge x.\,[\![A(x) \vee B(x)]\!]) \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]}$$

The goal $A(x) \vee B(x)$ is universal: it must be proved for arbitrary x.

The next resolution step can take place by lifting an object-rule over the universal variable x. This lifting principle ($\bigwedge$-lifting) resembles the one for lifting an object-rule over assumptions ($\Longrightarrow$-lifting). The case of an implication with one free variable c and one antecedent $\mathcal{A}(c)$ is

$$\frac{\mathcal{A}(c) \Longrightarrow \mathcal{B}(c)}{(\bigwedge x.\mathcal{A}(c'(x))) \Longrightarrow \bigwedge x.\mathcal{B}(c'(x))}$$

The implication $\mathcal{A}(c) \Longrightarrow \mathcal{B}(c)$ reduces $\mathcal{B}(c)$ to $\mathcal{A}(c)$ for any expression c, including one of the form $c'(x)$. To show $\mathcal{B}(c'(x))$ for all x, it suffices to show $\mathcal{A}(c'(x))$ for all x. The formal derivation in higher-order logic is

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{A}(c) \Longrightarrow \mathcal{B}(c)}{\bigwedge z.\,\mathcal{A}(z) \Longrightarrow \mathcal{B}(z)}}{\mathcal{A}(c'(x)) \Longrightarrow \mathcal{B}(c'(x))} \qquad \cfrac{[\bigwedge x.\mathcal{A}(c'(x))]}{\mathcal{A}(c'(x))}}{\mathcal{B}(c'(x))}}{\bigwedge x.\mathcal{B}(c'(x))}}{(\bigwedge x.\mathcal{A}(c'(x))) \Longrightarrow \bigwedge x.\mathcal{B}(c'(x))}$$

Lifting generalizes in the obvious way to allow any number of universal variables $x_1, \ldots, x_k$ (abbreviated $\vec{x}$), free variables $c_1, \ldots, c_n$ and antecedents $\mathcal{A}_1, \ldots, \mathcal{A}_m$:

$$\frac{[\mathcal{A}_1(c_1, \ldots, c_n), \ldots, \mathcal{A}_m(c_1, \ldots, c_n)] \Longrightarrow \mathcal{A}(c_1, \ldots, c_n)}{[\bigwedge \vec{x}.\,\mathcal{A}_1(c_1'(\vec{x}), \ldots, c_n'(\vec{x})), \ldots, \bigwedge \vec{x}.\,\mathcal{A}_m(c_1'(\vec{x}), \ldots, c_n'(\vec{x}))] \Longrightarrow \bigwedge \vec{x}.\,\mathcal{A}(c_1'(\vec{x}), \ldots, c_n'(\vec{x}))}$$

Examples: The $\vee$-introduction axiom has free variables A and B. Lifting it over the universal variable x produces a theorem with new free variables A' and B', functions of x:

$$\frac{[\![A]\!] \Longrightarrow [\![A \vee B]\!]}{(\bigwedge x.[\![A'(x)]\!]) \Longrightarrow \bigwedge x.[\![A'(x) \vee B'(x)]\!]}$$

The $\exists$-introduction axiom has free variables A' and t. Lifted over x, it obtains free variables A'' and t':

$$\frac{[\![A'(t)]\!] \Longrightarrow [\![\exists x.'A(x)]\!]}{(\bigwedge x.[\![A''(x,t'(x))]\!]) \Longrightarrow \bigwedge x.[\![\exists y.A''(x,y)]\!]}$$

To avoid collision, the inner bound variable is renamed from x to y. Renaming a bound variable does not affect the meaning of an expression.

We can now resume the backwards proof of $\forall x.A(x) \vee B(x)$ from $\forall x.A(x)$. Recall that the proof state after the first step was

$$(\bigwedge x.\,[\![A(x) \vee B(x)]\!]) \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]$$

Resolution with the $\bigwedge$-lifted version of the $\vee$-introduction axiom simplifies the general subgoal, $A(x) \vee B(x)$:

$$\frac{(\bigwedge x.[\![A_2'(x)]\!]) \Longrightarrow \bigwedge x.[\![A_2'(x) \vee B_2'(x)]\!] \qquad (\bigwedge x.\,[\![A(x) \vee B(x)]\!]) \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]}{(\bigwedge x.\,[\![A(x)]\!]) \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]}$$

The instantiations are A_2' to A and B_2' to B.

Recall that when deriving a rule, its premises may be taken as assumptions. The assumption $[\![\forall x.A(x)]\!]$, by the axiom of $\forall$-elimination, proves $[\![A(t_3)]\!]$, where t_3 is a free variable for resolution. Lifting over x gives the theorem $\bigwedge x.\,[\![A(t_3'(x))]\!]$, where t_3' is a free function variable. In the final resolution, t_3' is instantiated to $\lambda x.x$, and so $t_3'(x) \equiv (\lambda x.x)(x) \equiv x$:

$$\frac{\bigwedge x.\,[\![A(t_3'(x))]\!] \qquad (\bigwedge x.\,[\![A(x)]\!]) \Longrightarrow [\![\forall x.A(x) \vee B(x)]\!]}{[\![\forall x.A(x) \vee B(x)]\!]}$$

This use of function variables is typical. In $\bigwedge x.\,A(t'(x))$, instantiating the function variable t' can replace the expression $t'(x)$ by an expression involving x. In $\bigwedge x.\,A(t)$ the variable t may not be replaced by an expression containing x free, for the quantifier would capture that free variable. In $\bigwedge xy.\,A(t'(x))$, instantiating t' can replace $t'(x)$ by an expression containing x free but not y. The general variables of the goal define a context. An expression like $t'(x)$, through the variables in its argument list, states precisely its dependence on the context.

6.2 Unification

Unification is essential for reasoning involving quantifiers. We prove that an algorithm runs in linear time by proving that if its input has size n then it runs in Kn seconds, for some K. We prove $a < b$ by proving $a < c$ and $c < b$ for some c. We use the fact $\forall x.P(x) \vee Q(x)$ for case analysis by saying that either $P(a)$ or $Q(a)$ is true, leaving a unspecified. Each of these examples involve goals containing unknowns: terms that must eventually be stated to complete the proof. In backwards proof, unification instantiates variables in goals. Sokolowski's proof of the soundness of Hoare's logic is an example of unification in practice [23].

The present approach to backwards proof, using derived rules, easily handles goals containing unknowns. We simply extend the resolution rule (2) to instantiate both its premises, the object-rule and the proof state. The new resolution rule is applicable when θ is a unifier of $\mathcal{A}$ and $\mathcal{B}$. Variables in goals, including the final goal, may be instantiated: $\mathcal{C}$ becomes $\mathcal{C}\theta$.

If $\mathcal{A}\theta \equiv \mathcal{B}\theta$ then

$$\frac{[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A} \qquad \mathcal{F} \Longrightarrow (\mathcal{H} \Longrightarrow \mathcal{B}) \Longrightarrow \mathcal{C}}{\mathcal{F}\theta \Longrightarrow [\mathcal{H}\theta \Longrightarrow \mathcal{A}_1\theta, \ldots, \mathcal{H}\theta \Longrightarrow \mathcal{A}_m\theta] \Longrightarrow \mathcal{C}\theta} \tag{3}$$

A resolution rule could incorporate $\bigwedge$-lifting, but how could we write it down? The examples continue to show the $\bigwedge$-lifting steps separately.

6.3 Basic examples involving quantifiers

To see how unification is used, try proving $\exists x. A(x) \vee B(x)$ from $\exists x. A(f(x))$. To exercise all the quantifier rules of first-order logic, prove $\forall x.\exists y.A(x,y)$ from $\exists y.\forall x.A(x,y)$, and understand the syntactic restrictions that make the converse impossible to prove. Here we will work two simpler examples.

Consider first-order logic with equality, where $=$ in particular is reflexive: we have the axiom $[\![t = t]\!]$. A proof of $\forall x.\exists y. x = y$ will be contrasted with an attempted proof of $\exists y.\forall x. x = y$, illustrating the effect of the provisos of the quantifier rules.

A successful proof

A first-order proof of $\forall x.\exists y. x = y$ is

$$\cfrac{\cfrac{\overline{x = x}}{\exists y. x = y}}{\forall x.\exists y. x = y}$$

The topmost inference is the reflexivity axiom.

The first resolution in the backwards proof of $\forall x.\exists y. x = y$ involves the $\forall$-introduction axiom, instantiating A'_1 to $\lambda x.\exists y. x = y$:

$$\frac{(\bigwedge x.[\![A'_1(x)]\!]) \Longrightarrow [\![\forall x.A'_1(x)]\!] \qquad [\![\forall x.\exists y. x = y]\!] \Longrightarrow [\![\forall x.\exists y. x = y]\!]}{(\bigwedge x.[\![\exists y. x = y]\!]) \Longrightarrow [\![\forall x.\exists y. x = y]\!]}$$

Next we resolve with the $\exists$-introduction axiom, after lifting it over the universal variable x:

$$\frac{(\bigwedge x.[\![A''_2(x, t'_2(x))]\!]) \Longrightarrow \bigwedge x.[\![\exists y.A''_2(x,y)]\!] \qquad (\bigwedge x.[\![\exists y.x = y]\!]) \Longrightarrow [\![\forall x.\exists y.x = y]\!]}{(\bigwedge x.[\![x = t'_2(x)]\!]) \Longrightarrow [\![\forall x.\exists y.x = y]\!]}$$

The variable A''_2 is instantiated to $\lambda xy. x = y$, so the normal form of $A''_2(x, t'_2(x))$ is $x = t'_2(x)$.

We can see that putting $\lambda x.x$ for t'_2 solves the subgoal, $\bigwedge x.[\![x = t'_2(x)]\!]$, by reflexivity of $=$. In the formal proof, this happens by resolving the proof state with the lifted form of the reflexivity axiom:

$$\frac{\bigwedge x.[\![t'_3(x) = t'_3(x)]\!] \qquad (\bigwedge x.[\![x = t'_2(x)]\!]) \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}{[\![\forall x.\exists y.\, x = y]\!]}$$

Consider the steps of higher-order unification [11,17]. The initial disagreement pair is

$$\langle \bigwedge x.[\![t'_3(x) = t'_3(x)]\!],\ \bigwedge x.[\![x = t'_2(x)]\!]\rangle$$

It reduces to the pairs $\langle \lambda x.t'_3(x), \lambda x.x\rangle$ and $\langle \lambda x.t'_3(x), \lambda x.t'_2(x)\rangle$. The first pair forces $t'_3(x)$ to be $\lambda x.x$; the second forces $t'_2(x)$ to be $\lambda x.x$. So the common instance is $\bigwedge x.[\![x = x]\!]$.

An unsuccessful proof

An attempt to prove $\exists y.\forall x.\, x = y$ in first-order logic is

$$\cfrac{\cfrac{x = t}{\forall x.\, x = t}}{\exists y.\forall x.\, x = y}$$

The topmost formula is unprovable since x and t must be distinct terms.

The first resolution in the attempted proof of $\exists y.\forall x.\, x = y$ involves the $\exists$-introduction axiom:

$$\frac{[\![A'_1(t_1)]\!] \Longrightarrow [\![\exists x.A'_1(x)]\!] \qquad [\![\exists y.\forall x.\, x = y]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}{[\![\forall x.\, x = t_1]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}$$

The subgoal contains a new variable, t_1.

Resolution with the $\forall$-introduction axiom gives

$$\frac{(\bigwedge x.[\![A'_2(x)]\!]) \Longrightarrow [\![\forall x.A'_2(x)]\!] \qquad [\![\forall x.\, x = t_1]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}{(\bigwedge x.[\![x = t_1]\!]) \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}$$

We are stuck. The subgoal $\bigwedge x.[\![x = t_1]\!]$ is false; no term t_1 is equal to everything. Resolution with the reflexivity axiom fails. The initial disagreement pair

$$\langle \bigwedge x.[\![t'_3(x) = t'_3(x)]\!],\ \bigwedge x.[\![x = t'_1]\!]\rangle$$

reduces to the pairs $\langle \lambda x.t'_3(x), \lambda x.x\rangle$ and $\langle \lambda x.t'_3(x), \lambda x.t'_1\rangle$. The first pair forces $t'_3(x)$ to be $\lambda x.x$, reducing the second to $\langle \lambda x.x, \lambda x.t'_1\rangle$, which has no unifier.

7 An alternative for quantifiers: Skolemization

Isabelle-86 does not use $\bigwedge$-lifting; it enforces parameter provisos like 'x is not free in Γ' literally. The $\forall$-introduction rule of the first-order sequent calculus is

$$\frac{\Gamma \vdash A(y)}{\Gamma \vdash \forall x.A(x)}$$

subject to the proviso that y is not free in Γ or A. Isabelle-86 reduces the goal $\Gamma \models \forall x.A(x)$ to the subgoal $\Gamma \models A(y)$, representing the proviso as a directed acyclic graph with arcs from y to Γ and A. The graph grows as schematic variables of Γ and A become instantiated. Isabelle-86 enforces the provisos by forbidding instantiations that would introduce a cycle into the graph. During resolution, it renames the variables of one of the premises to prevent clashes.

Here is a basic observation about universal quantifiers and implication. The rule (on the left) is valid; the implication (on the right) is not:

$$\frac{\mathcal{A}(y)}{\bigwedge x.\mathcal{A}(x)} \qquad\qquad \mathcal{A}(y) \Longrightarrow \bigwedge x.\mathcal{A}(x)$$

To eliminate universal variables in goals, we must replace y by a special expression that makes the implication valid.

7.1 Hilbert's ϵ-operator

Church's original higher-order logic includes Hilbert's ϵ-operator where $\epsilon x.\mathcal{A}(x)$ is an expression provided $\mathcal{A}(x)$ is a proposition. If $\mathcal{A}(b)$ is true for some b then $\epsilon x.\mathcal{A}(x)$ is b. Otherwise $\epsilon x.\mathcal{A}(x)$ has an arbitrary value: all types are non-empty, so a member can be chosen using the Axiom of Choice. An axiom for the ϵ-operator is

$$\bigwedge x.\mathcal{B}(x) \Longrightarrow \mathcal{B}(\epsilon x.\mathcal{B}(x))$$

Suppose that we extend our fragment of higher-order logic with classical negation. Then putting $\neg\mathcal{A}(x)$ for $\mathcal{B}(x)$ gives

$$\bigwedge x.\neg\mathcal{A}(x) \Longrightarrow \neg\mathcal{A}(\epsilon x.\neg\mathcal{A}(x))$$

Taking the contrapositive, and pushing the $\bigwedge x$ inwards, gives

$$\mathcal{A}(\epsilon x.\neg\mathcal{A}(x)) \Longrightarrow \bigwedge x.\mathcal{A}(x)$$

So our special expression is $\epsilon x.\neg\mathcal{A}(x)$, which is a value chosen to falsify $\mathcal{A}$, if possible.

The expression $\epsilon x.\neg\mathcal{A}(x)$ contains the same free variables as $\bigwedge x.\mathcal{A}(x)$, as can be seen when it is instantiated. For example, a theorem representing $\forall$-introduction in first-order logic is

$$\bigwedge A.\ [\![A(\epsilon x.\neg[\![A(x)]\!])]\!] \Longrightarrow [\![\forall x.A(x)]\!]$$

where the variable A is free in $\epsilon x.\neg[\![A(x)]\!]$ but bound in the surrounding expression. Specializing A to $\lambda x.a(x) = 0$ gives the theorem

$$[\![a(\epsilon x.\neg[\![a(x) = 0]\!]) = 0]\!] \Longrightarrow [\![\forall x.a(x) = 0]\!]$$

Specializing A to $\lambda x.B(x) \supset C(x)$ gives the theorem

$$[\![B(\epsilon x.\neg[\![B(x) \supset C(x)]\!]) \supset C(\epsilon x.\neg[\![B(x) \supset C(x)]\!])]\!] \Longrightarrow [\![\forall x.B(x) \supset C(x)]\!]$$

In the resulting theorems, the expression $\epsilon x.\neg[\![A(x)]\!]$ produces two different expressions

$$\epsilon x.\neg[\![a(x)=0]\!] \qquad \text{and} \qquad \epsilon x.\neg[\![B(x)\supset C(x)]\!]$$

7.2 Replacing Hilbert's ϵ by Skolem constants

The ϵ-operator is impractical because the expression $\epsilon x.\neg\mathcal{A}(x)$ grows to enormous size. Is there something similar and less unwieldy? Consider adding to HOL the axiom scheme

$$\mathcal{A}(\mathbf{y}_{\mathcal{A}}) \Longrightarrow \bigwedge x.\mathcal{A}(x) \tag{4}$$

The intent is that, for each instance of this scheme, $\mathbf{y}_{\mathcal{A}}$ is a *unique* constant not free in $\mathcal{A}$. In $\mathbf{y}_{\mathcal{A}}$ the variable $\mathcal{A}$ can be seen as part of the name, though an implementation need not take this literally. Isabelle-86 generates a unique name and associates it with the free variables of $\mathcal{A}$ through the graph mechanism mentioned earlier.

Since $\mathbf{y}_{\mathcal{A}}$ is a constant, it can only occur free, so its free variables must not be bound in the surrounding expression. The generalization rule ($\bigwedge$-introduction) must be accordingly restricted. Recall the instantiation rule, $\mathcal{B}(x)/\mathcal{B}(a)$. Previously derived by generalization and specialization, it must be taken as primitive if we adopt Skolemization. In Skolem constants of the premise, every free occurrence of x is replaced by a, an expression of the same type.

The Skolemized version of the $\forall$-introduction rule is

$$[\![A(\mathbf{y}_A)]\!] \Longrightarrow [\![\forall x.A(x)]\!]$$

This cannot be generalized over A because A is free in the constant $\mathbf{y}_A$. However, A may be instantiated. Instantiating the theorem instantiates the free variables of A, creating a new Skolem constant. Instantiating A to $\lambda x\,.\,a(x)=0$ yields

$$[\![a(\mathbf{y}_a)=0]\!] \Longrightarrow [\![\forall x\,.\,a(x)=0]\!]$$

Instantiating A to $\lambda x.B(x)\supset C(x)$ yields

$$[\![B(\mathbf{y}_{B,C})\supset C(\mathbf{y}_{B,C})]\!] \Longrightarrow [\![\forall x.B(x)\supset C(x)]\!]$$

The resulting theorems contain different constants $\mathbf{y}_a$ and $\mathbf{y}_{B,C}$, as though $\mathbf{y}_A$ were renamed in every use.

7.3 Sample proofs using Skolem constants

Let us return to the two quantifier examples from Section 6.3: the proof of $\forall x.\exists y.\,x=y$ and failed proof of $\exists y.\forall x.\,x=y$. These examples illustrate how provisos of quantifier rules are enforced; here Skolem constants and lifting differ.

A successful proof

The first resolution in the backwards proof of $\forall x.\exists y.\, x = y$ involves the axiom of $\forall$-introduction:

$$\frac{[\![A_1'(\mathbf{x}_{A_1'})]\!] \Longrightarrow [\![\forall x.A_1'(x)]\!] \qquad [\![\forall x.\exists y.\, x = y]\!] \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}{[\![\exists y.\, \mathbf{x} = y]\!] \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}$$

The constant $\mathbf{x}$ in the conclusion has no subscript since A_1' is assigned $\lambda x.\exists y.\, x = y$, which has no free variables.

Next we resolve with the $\exists$-introduction axiom:

$$\frac{[\![A_2'(t_2)]\!] \Longrightarrow [\![\exists x.A_2'(x)]\!] \qquad [\![\exists y.\, \mathbf{x} = y]\!] \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}{[\![\mathbf{x} = t_2]\!] \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}$$

The function variable A_2' is instantiated to $\lambda y.\, \mathbf{x} = y$, so the normal form of $A_2'(t_2)$ is $\mathbf{x} = t_2$.

Resolving the proof state with the reflexivity axiom sets t_2 to $\mathbf{x}$, completing the proof:

$$\frac{[\![t_3 = t_3]\!] \qquad [\![\mathbf{x} = t_2]\!] \Longrightarrow [\![\forall x.\exists y.\, x = y]\!]}{[\![\forall x.\exists y.\, x = y]\!]}$$

An unsuccessful proof

The first resolution in the attempted proof of $\exists y.\forall x.\, x = y$ is the same as in Section 6.3:

$$\frac{[\![A_1'(t_1)]\!] \Longrightarrow [\![\exists x.A_1'(x)]\!] \qquad [\![\exists y.\forall x.\, x = y]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}{[\![\forall x.\, x = t_1]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}$$

Resolution with the $\forall$-introduction axiom gives

$$\frac{[\![A_2'(\mathbf{x}_{A_2'})]\!] \Longrightarrow [\![\forall x.A_2'(x)]\!] \qquad [\![\forall x.\, x = t_1]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}{[\![\mathbf{x}_{t_1} = t_1]\!] \Longrightarrow [\![\exists y.\forall x.\, x = y]\!]}$$

The instantiation of A_2' is $\lambda x.\, x = t_1$. Its free variable, t_1, is the subscript of the constant $\mathbf{x}_{t_1}$ in the conclusion. The subgoal $[\![\mathbf{x}_{t_1} = t_1]\!]$ cannot be solved because the instantiation of t_1 to $\mathbf{x}_{t_1}$ would be circular; these expressions are not unifiable.

7.4 Lifting versus Skolemization

Let us compare $\bigwedge$-lifting with Skolem constants by considering the abstract form of a derivation. In each case, the final goal is $\mathcal{B}$ and each partial proof involves a single subgoal.

lifting	*Skolem constants*
$\mathcal{A}(a) \Longrightarrow \mathcal{B}$	$\mathcal{A}(a) \Longrightarrow \mathcal{B}$
$(\bigwedge x.\mathcal{A}_1(a, x)) \Longrightarrow \mathcal{B}$	$\mathcal{A}_1(a, \mathbf{x}_a) \Longrightarrow \mathcal{B}$
$(\bigwedge x.\mathcal{A}_2(a, x, b(x))) \Longrightarrow \mathcal{B}$	$\mathcal{A}_2(a, \mathbf{x}_a, b) \Longrightarrow \mathcal{B}$
$(\bigwedge xy.\mathcal{A}_3(a, x, b(x), y)) \Longrightarrow \mathcal{B}$	$\mathcal{A}_3(a, \mathbf{x}_a, b, \mathbf{y}_{a,b}) \Longrightarrow \mathcal{B}$
$(\bigwedge xy.\mathcal{A}_4(a, x, b(x), y, c(x, y))) \Longrightarrow \mathcal{B}$	$\mathcal{A}_4(a, \mathbf{x}_a, b, \mathbf{y}_{a,b}, c) \Longrightarrow \mathcal{B}$

In the sequence of proof states new variables appear one by one: the free variable a, the universal variable x, the free variable b, the universal variable y, and the free variable c. Consider the final theorem:

lifting	*Skolem constants*
The variable a can not be assigned an expression containing x or y free because bound variables can not be captured.	The variable a can not be assigned an expression containing $\mathbf{x}_a$ or $\mathbf{y}_{a,b}$ because a is free in these constants.
The variable $b(x)$ can be assigned an expression containing x but not y.	The variable b can not be assigned an expression containing $\mathbf{y}_{a,b}$, which contains b.
The variable $c(x,y)$ can be assigned an expression containing x or y.	The variable c can be assigned an expression containing $\mathbf{x}_a$ or $\mathbf{y}_{a,b}$, for c is free in neither.

Skolemization raises many questions. When is the name of a Skolem constant significant? What is the scope of a Skolem constant? Does the axiom scheme (4) entail classical logic or the Axiom of Choice? It is a conservative extension if every proof of $\mathcal{A}(\mathbf{y}_\mathcal{A})$ can be transformed to a proof of $\bigwedge x.\mathcal{A}(x)$, which is the converse of Herbrand's theorem [6], for higher-order logic. For further discussion, see Miller [16].

With $\bigwedge$-lifting the status of universal variables is clear. The name of a universal variable in a goal is insignificant, by α-conversion. The scope of a universal variable is limited to its goal. In

$$(\bigwedge x.\mathcal{A}(x)) \Longrightarrow (\bigwedge y.\mathcal{B}(y)) \Longrightarrow \mathcal{C}$$

the scope of x is the first goal and that of y is the second; neither can occur in $\mathcal{C}$. Lifting does not extend the original logic.

A drawback of $\bigwedge$-lifting is the increased number of function variables, which places increased demands on higher-order unification. But higher-order unification is already essential in the present approach.

8 An implementation

Isabelle-86 is concerned with object-rules of the form $[\mathcal{A}_1, \ldots, \mathcal{A}_m] \Longrightarrow \mathcal{A}$. Implication cannot be nested such that the premises $\mathcal{A}_i$ are themselves rules. Quantifiers are not available, but schematic variables and Skolemization handle things like

$$\bigwedge x.(\bigwedge y.\mathcal{A}(x,y)) \Longrightarrow \mathcal{B}(x)$$

Schematic variables allow rules to be instantiated. Skolem constants allow rules with universal premises. I have outlined the ideas behind Isabelle-86 elsewhere [17].

Below the level of rules, Isabelle is concerned with terms, substitution, normalization, higher-order unification, and also parsing and printing. Higher levels are concerned with tactics and tacticals. Above basic Isabelle come the object-logics with their special inference mechanisms.

I implemented a prover based on higher-order logic by modifying Isabelle-86. Only the level of rules needed substantial change; the lower levels were slightly modified and the higher levels hardly at all. The natural deduction rules of Section 2.4 were represented by the corresponding sequent calculus, so theorems have the form $\mathcal{G} \models \mathcal{A}$. While object-rules have visible structure, meta-rules are represented like in LCF: by functions. The resolution rules were implemented directly; deriving them from the primitive rules would be painfully slow.

The $\Longrightarrow$-lifting rule has not been implemented, so Isabelle does not yet support natural deduction. But sequent calculi may be preferable. They are the natural system for classical logic [20, page 245]. And in backwards proof they allow the deletion of needless assumptions. Compare the conjunction and quantifier rules:

$$\frac{\Gamma, A, B \models \Delta}{\Gamma, A \wedge B \models \Delta} \qquad\qquad \frac{\Gamma, \forall x.A(x), A(t) \models \Delta}{\Gamma, \forall x.A(x) \models \Delta}$$

In backwards proof, the assumption $A \wedge B$ is useless in the subgoal, while further instances of $\forall x.A(x)$ may be required. The corresponding rules in natural deduction do not make this distinction.

The Skolem constants of Isabelle-86 have been abandoned in favor of $\bigwedge$-lifting, but schematic variables have been kept. Unlike ordinary variables, they can be instantiated without checking the assumptions. The meta-rules guarantee that the HOL sequent $\mathcal{G} \models \mathcal{A}$ has schematic variables only in $\mathcal{A}$. The instantiation rule is

$$\frac{\mathcal{G} \models \mathcal{A}}{\mathcal{G} \models \mathcal{A}[a_1/x_1, \ldots, a_k/x_k]}$$

where $x_1, \ldots, x_k$ are schematic variables — and therefore do not appear in the assumptions $\mathcal{G}$.

The sample logics of Isabelle-86 were implemented. Many of the standard examples for Intuitionistic Type Theory worked unchanged. The largest Intuitionistic Type Theory example develops elementary number theory up to the theorem

$$a \bmod b + (a/b) \times b = a$$

These proofs rely heavily on a rewriting tactic that works by resolution with the equality rules of Type Theory.

A sequent calculus for intuitionistic first-order logic was made to work within a week. Later, a classical sequent calculus was added. Most of the first-order logic examples also worked, including automatic proof procedures based on an encoding of associative unification via higher-order unification. These procedures are not complete (they use quantified formulae only once) but can automatically prove many theorems in either intuitionistic or classical first-order logic.

A few examples involved ambiguous use of a rule, where higher-order unification produced multiple unifiers. Lifting caused the unifiers to be produced in a different order. Changing the unification algorithm to reverse the order of projections [17] allowed these examples to work again.

Equality ($\equiv$) handles constraints and definitions. If higher-order unification returns flex-flex disagreement pairs [11,17], say $\langle a_1, b_1\rangle, \ldots, \langle a_m, b_m\rangle$, then resolution imposes equality constraints on the conclusion, returning a theorem of the form

$$a_1 \equiv b_1 \Longrightarrow \cdots \Longrightarrow a_m \equiv b_m \Longrightarrow C$$

The old definition mechanism has been replaced by one using equality reasoning. A constant is defined by an equation like $K \equiv a$, allowing the replacement of K by a when desired. The examples involving definitions have been reworked.

Isabelle is written in Standard ML. I took the opportunity to adopt ML modules, and spent much time chasing compiler bugs that came to light. The new version runs slightly slower than the old: partly due to lifting, and partly because modules add a layer of indirection.

Producing the implementation was easy. This confirms that HOL is a straightforward extension of the meta-logic of Isabelle-86.

9 Related work

Related work includes AUTOMATH [12], the Calculus of Constructions [5], the Edinburgh Logical Framework (LF) [9], and the work of Martin-Löf [15]. These calculi have much in common but differ on practical and philosophical points. They have different objectives: AUTOMATH and the Constructions, to formalize mathematical reasoning; the others, to be a general representation for logics.

The other systems are all higher-order λ-calculi using the interpretation of *propositions-as-types* [13]. A proposition is represented by a type; proving the proposition A means constructing a proof object of type A. Proof objects grow with the length of the proof; storing them may be impractical. Nobody knows whether proof objects of an arbitrary logic have any use, though Constable et al. use those of Martin-Löf's Type Theory as a functional programming language [4].

Propositions-as-types reflects the intuitionistic interpretation of the logical connectives. The implication $A \Longrightarrow B$ and the quantification $\bigwedge x \in A \,.\, B(x)$ are both expressed by the product $\Pi x \in A \,.\, B(x)$. This view eliminates the difference between $\bigwedge$-lifting and $\Longrightarrow$-lifting. In HOL the constant $\wedge$ has a function type, while the axiom for $\wedge$-introduction uses implication:

$$\wedge : \mathit{form} \rightarrow (\mathit{form} \rightarrow \mathit{form}) \qquad\qquad [\![A]\!] \Longrightarrow ([\![B]\!] \Longrightarrow [\![A \wedge B]\!])$$

Propositions-as-types represents implication by a function type, identifying $\rightarrow$ and $\Longrightarrow$.

The Edinburgh group have formalized a variety of unusual logics in their LF [3]. Their formalization of modal logic reflects its 'possible worlds' semantics. To

$$\dfrac{\dfrac{\forall x.A(x)}{A(x)}}{\exists x.A(x)} \qquad\qquad \dfrac{\dfrac{\forall x.A(x)}{A(0)}}{\exists x.A(x)}$$

$$(a) \qquad\qquad\qquad (b)$$

Figure 2: Two proofs of $\exists x.A(x)$ from $\forall x.A(x)$

handle the complex proviso of the necessitation rule, they are led to introduce two forms of assertion: truth in this world and truth in a frame of related worlds. We see that a meta-logic formalizes the *semantic* justifications of the object-rules; it is often impractical to formalize the rules literally.

Propositions-as-types requires a rich type structure, including indexed families of types. These easily handle typed object-logics [9]: the type *term*(T) might represent the set of terms of object-type T. The representation of such a logic in HOL involves the type *term* of all terms, including terms having no legal object-type; propositions express object-level type checking rules. On the other hand, the representation of first-order logic given above is similar to that of Martin-Löf [15] and the Edinburgh group [9].

A subtle difference: HOL assumes that all types are non-empty, for an interpretation where $\mathcal{A}$ is false and the type of x is empty falsifies the theorem $(\bigwedge x.\mathcal{A}) \Longrightarrow \mathcal{A}$. Under propositions-as-types, some types must be empty, namely those representing false propositions. Standard first-order logic assumes a non-empty universe, and so $\forall x.A(x)$ implies $\exists x.A(x)$. HOL can represent the standard proof, Figure 2(a). Avron et al. [3] suggest that deriving this in the LF formalization of first-order logic requires a constant, say 0. The LF proof presumably represents that of Figure 2(b).

Schroeder-Heister has devised a system of natural deduction with rules of higher levels [21,22]. Assumptions may be rules, not just formulae, and can be discharged in the application of another rule. His system is complicated but seems to use the same primitive concepts as everyone else: forms of meta-implication and meta-quantification.

My theorem prover is designed, above all, to make full use of unification. Huet's unification algorithm for higher-order logic [11] has been shown to be useful by myself and others [2,17]. Most of the other calculi have bound variables in types, complicating the unification problem. Harper et al. claim to 'have defined a logic-independent search space that generalizes Paulson's higher-order resolution'. Lacking a unification algorithm, it is unclear what the authors could mean by this.

In higher-order logic the other connectives can be defined using $\Longrightarrow$ and $\bigwedge$ by quantification over propositions, a standard construction [19]. The disjunction

$\mathcal{A} \vee \mathcal{B}$ is equivalent to the proposition

$$\bigwedge \mathcal{C}.(\mathcal{A} \Longrightarrow \mathcal{C}) \Longrightarrow (\mathcal{B} \Longrightarrow \mathcal{C}) \Longrightarrow \mathcal{C}$$

This is an *impredicative* proposition: it refers to all propositions, including itself. A similar example is Leibniz's definition of equality: $x = y$ means y has every property of x, namely $\bigwedge \mathcal{A}.\mathcal{A}(x) \Longrightarrow \mathcal{A}(y)$. Careless use of impredicative propositions can lead to contradiction [24], but HOL allows them and is nonetheless consistent. HOL can be made predicative by forbidding bound variables whose type involves *prop*. The resulting system is much weaker but can still serve as a meta-logic: to represent object-logics requires only quantification over *object*-formulae. Its theory of proof normalization is much simpler, essentially the same as first-order logic. Most of the other calculi are predicative and intuitionistic.

Acknowledgements. Thierry Coquand has made many invaluable contributions, especially in normalization problems. Thanks also to Peter Dybjer, Martin Hyland, Brian Monahan, and the Edinburgh group. Dave Matthews has continued to support Poly/ML. Funding and equipment were provided by the SERC/Alvey grant GR/E0355.7.

References

[1] P. B. Andrews, *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof* (Academic Press, 1986).

[2] P. B. Andrews, D. A. Miller, E. L. Cohen, F. Pfenning, Automating higher-order logic, in: W. W. Bledsoe and D. W. Loveland, editors, *Automated Theorem Proving: After 25 Years* (American Mathematical Society, 1984), pages 169–192.

[3] A. Avron, F. A. Honsell, I. A. Mason, Using typed lambda calculus to implement formal systems on a machine, Draft report, University of Edinburgh (1987).

[4] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panagaden, J. T. Sasaki, S. F. Smith, *Implementing Mathematics with the Nuprl Proof Development System* (Prentice-Hall, 1986).

[5] Th. Coquand, G. Huet, Constructions: a higher order proof system for mechanizing mathematics, *in*: B. Buchberger, editor, *EUROCAL '85: European Conference on Computer Algebra*, Volume 1: *Invited lectures* (Springer, 1985), pages 151–184.

[6] J. H. Gallier, *Logic for Computer Science: Foundations of Automatic Theorem Proving* (Harper & Row, 1986).

[7] M. J. C. Gordon, HOL: A proof generating system for higher-order logic, *in*: G. Birtwistle, P. A. Subrahmanyam, editors, *VLSI Specification, Verification and Synthesis* (Kluwer Academic Publishers, 1987, in press).

[8] Ph. de Groote, How I spent my time in Cambridge with Isabelle, Report RR 87-1, Unité d'informatique, Université Catholique de Louvain, Belguim (1987).

[9] R. Harper, F. Honsell, G. Plotkin, A Framework for Defining Logics, *Logic in Computer Science* (IEEE, Proceedings, 1987), pages 194–204.

[10] J. R. Hindley and J. P. Seldin, *Introduction to Combinators and λ-calculus* (Cambridge University Press, 1986).

[11] G. P. Huet, A unification algorithm for typed λ-calculus, *Theoretical Computer Science* 1 (1975), pages 27–57.

[12] L. S. van Benthem Jutting, *Checking Landau's 'Grundlagen' in the* AUTOMATH *system*, PhD Thesis, Technische Hogeschool, Eindhoven (1977).

[13] P. Martin-Löf, Constructive mathematics and computer programming, *in*: C. A. R. Hoare, J. C. Shepherdson, editors, *Mathematical Logic and Programming Languages* (Prentice-Hall, 1985), pages 167–184.

[14] P. Martin-Löf, On the meanings of the logical constants and the justifications of the logical laws, Report, Department of Mathematics, University of Stockholm (1986).

[15] P. Martin-Löf, Amendment to intuitionistic type theory, Notes obtained from P. Dybjer, Computer Science Department, Chalmers University, Gothenburg (1986).

[16] D. A. Miller, *Proofs in Higher-order Logic*, PhD. thesis, Carnegie-Mellon University (1983). Also report MS-CIS-83-37, Department of Computer and Information Science, University of Pennsylvania.

[17] L. C. Paulson, Natural deduction as higher-order resolution, *Journal of Logic Programming* 3 (1986), pages 237–258.

[18] L. C. Paulson, *Logic and Computation: Interactive Proof with Cambridge LCF* (Cambridge University Press, 1987, in press).

[19] D. Prawitz, *Natural Deduction: A Proof-theoretical Study* (Almquist and Wiksell, 1965).

[20] D. Prawitz, Ideas and results in proof theory, *in*: J. E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium* (North-Holland, 1971), pages 235–308.

[21] P. Schroeder-Heister, A natural extension of natural deduction, *Journal of Symbolic Logic* 49 (1984), pages 1284–1300.

[22] P. Schroeder-Heister, Generalized rules for quantifiers and the completeness of the intuitionistic operators &, $\vee$, $\supset$, $\perp$, $\forall$, $\exists$, *in*: M. M. Richter et al., editors, *Logic Colloquium '83* (Springer Lecture Notes in Mathematics 1104, 1984).

[23] S. Sokołowski, Soundness of Hoare's logic: an automatic proof using LCF, *ACM Transactions on Programming Languages and Systems* 9 (1987), pages 100–120.

[24] A. N. Whitehead, B. Russell, *Principia Mathematica* (Paperback edition to *56, Cambridge University Press, 1962).