Adding metatheoretic facilities to first-order theories

David Basin and Seán Matthews
Max-Planck-Institut für Informatik,
Im Stadtwald, D-66123 Saarbrücken, Germany
{basin,sean}@mpi-sb.mpg.de
phone: +49 681 302-5363 (fax: 302-5401)

November 13, 1995

Abstract

Generic proof systems like *Isabelle* provide some limited but useful metatheoretic facilities for declared logics; in particular, users can prove simple derived rules and also 'solve' formulae that contain metavariables — a technique useful for, e.g., program synthesis. We show how an arbitrary first order theory can be conservatively extended to provide similar facilities, without a supporting metatheory, and examine what the limitations of this approach are.

Keywords: computer assisted proof, logical frameworks, metatheoretic extensibility, second-order logic, proof theory.

1 Introduction

Using a generic proof development system like Isabelle [12] to prove theorems of a logic is sometimes easier than using a custom-built prover. This is because the generic system contains a *metalogic* in which *object logics* are encoded and this may allow extra methods of proof construction.

In this paper we look at two examples of this. First, in the metalogic, users can derive new rules or lemmas for the object logic and specializations of these can be applied later, during proof construction. Deriving these is not always possible in the object logic itself — for example, first-order logic (FOL) cannot express rules that contain variables ranging over first-order formulae. Second, if the generic proof system supports metavariables, e.g. Isabelle, we can delay commitment during proof, i.e. we prove that a formula $P(\alpha)$ has some desired property, without having to give α explicitly at the beginning of the proof building process (so that the proof and the formula being proved are developed together). This delaying of commitment can make it easier to construct proofs, since we can wait until we have more detailed information about what α must be, before we have to decide on the details of its structure.

Our contribution is to show that, rather than using a system like Isabelle, these kinds of extensibility can be directly added to an arbitrary first-order theory by conservatively extending it with quantifiers over formulae; that is, without recourse to an independent metatheory. This can be seen as a proof theoretic investigation into aspects of the metatheory of various generic proof systems and how they can be simulated in weak extensions of first-order theories.

While we hope our contribution has conceptual value we are not arguing for a change in practice: there are many advantages in using generic development systems that we do not address here. For example, systems based on type theory offer capabilities of representing proofs as terms in the logic which are built from constructive derivations [1] and the metalogic may be well suited for proof search which enables automated theorem proving for encoded object logics [13, 15].

Before providing technical details, we further motivate the two kinds of metatheoretic extensions we would like to capture, which are related to universally and existentially quantified variables in the metalogic.

1.1 Universally quantified metavariables

In theorem proving, we often encounter instances of the same pattern of reasoning. For instance, given a goal $\neg\exists x(p(x))$ we might want to simplify this using the fact that

$$\text{for any formula } \alpha, \text{ if } \forall x \neg\alpha \text{ then } \neg\exists x\alpha \tag{1}$$

to move the negation through the quantifier. However this transformation is not usually given as a rule or axiom of first-order logic, even though it is provable for each particular instance. If we do not want to type in the primitive derivation each time we need it, what is the most convenient way to make this fact available?

A proof development system often contains a *tactic language* or a *lemma facility* and these are used to capture certain patterns of reasoning. Tactics are the most primitive solution: instead of having to enter the steps of the derivation by hand each time, the user can write a program to perform the steps automatically. This way, the same primitive steps are performed (which guarantees correctness) but reliably and quickly by the machine. This, however, still takes time, and a tactic by its nature is a procedure which is often more difficult to understand than a declarative statement like (1).

A lemma facility does not have the problems of a tactic language: instead of building a proof each time, the user can store proven theorems to be reused as needed; thus there is no problem with speed (the theorem is already there to be used) or in understanding which theorem a tactic constructs. Unfortunately, a lemma cannot capture (1), because this cannot be written in FOL: we are not able to formalize the phrase 'for any formula α', because we cannot quantify over formulae.

If we could do this we could prove a single theorem and store as a lemma to be instantiated as necessary. But we would need something like a higher-order logic, in which we could write and prove (1) as

$$\forall \alpha (\forall x \alpha(x) \rightarrow \neg \exists x \neg \alpha(x)). \tag{2}$$

Unfortunately, higher-order logics are much more complex than first-order ones, and introduce new difficulties, even as they solve one. For instance many theorem proving techniques that work well in FOL either break in a higher-order setting or become much more inefficient (e.g. unification). More seriously, even when we restrict ourself to reasoning about sentences in our original language, we may find that more formulae become provable: the new, more expressive, theory may not be conservative over the original.

Now consider what we can do if we have quantifiers in the metatheory; we are able to formalize (1) precisely, as[1]

$$(\alpha)((\vdash \forall x \alpha) \Rightarrow (\vdash \neg \exists x \neg \alpha)). \tag{3}$$

This is a formula in the metalogic defining a derived rule of FOL — the outside (α) quantifies over formulae in the object logic, but does it in the metalogic — and we can prove such a formula in the metalogic, and store it as a lemma of that logic. Then we can instantiate it as necessary for use in particular derivations in the object logic. Further, this way we do not modify the object theory, so there is no risk of proving something that is not a theorem of the original logic. Several variations on this approach have been developed: Goguen [7], for instance, shows how, for the algebraic system OBJ, we get the equivalent of the metatheoretic facilities that we would like to have by allowing theories to be schematic; Boyer et al. [3] argue for a similar kind of schematic theorem facility in their *Nqthm* theorem prover.

1.2 Existentially quantified metavariables

In a system with an explicit metatheory, however, we are able to do something that neither Goguen nor Boyer and Moore consider. In the Isabelle system we can use higher-order unification to compose rules and axioms, and, like in (3), we are allowed metavariable 'holes' in object logic formulae. In (3) the metavariable is in a lemma, and universally quantified, but we can equally have metavariables in the formula we are trying to prove, and

[1] We use $(\alpha)A$ and $(\mathbf{E}\alpha)A$ to denote universal and existential quantification at the metalevel, and $\Rightarrow$ to indicate metalevel implication.

there is no obligation to quantify them universally. If we do this, then, as we build a proof, unification can instantiate metavariables with object-level syntax as necessary to ensure that we always have a valid proof schema. As a trivial example, we might try to prove

$$\vdash (P(a) \land Q(b) \to R(a,b)) \leftrightarrow \alpha \tag{4}$$

where we want the metavariable hole α on the right to be the implicational form of the formula on the left. We can unify (computing instances for metavariables) this against the bi-implication in the already proven lemma

$$(\beta)\,(\gamma)\,(\epsilon)(\vdash ((\beta \land \gamma) \to \epsilon) \leftrightarrow (\beta \to \gamma \to \epsilon))$$

and as a result α is instantiated to $P(a) \to Q(b) \to R(a,b)$. In [2] we use proof-theoretic techniques to show how this use of proofs with metavariables over formulae can be applied to problems in program synthesis. In particular, we can reason about the relationship between a specification S and a metavariable α and use Isabelle to find an appropriate instantiation of α so that the relationship (say, equivalence) is provable in the theory; e.g. we prove theorems like

$$\forall x_1, \ldots, x_n (S(x_1, \ldots, x_n) \leftrightarrow \alpha(x_1, \ldots, x_n)) \tag{5}$$

where S is a specification with parameters $x_1, \ldots, x_n$ and α is a metavariable which, at the end, is instantiated with a formula, representing a logic program that is a recursive version of same. Other applications of the use of metavariables for synthesis are discussed in Clement [4] and Hesketh [8].

Again there is a close relationship between what we are doing here and goals existentially quantified over formulae; i.e. (5) could also be written as

$$(\mathbf{E}\alpha)(\vdash \forall x_1, \ldots, x_n (S(x_1, \ldots, x_n) \leftrightarrow \alpha(x_1, \ldots, x_n))), \tag{6}$$

provided the **E** is taken to be constructive in the sense that we can find some formula A for α that satisfies the interior of the formula. (We should point out that there are actually no existential quantifiers in Isabelle, but metavariables in goals allow us to behave as if there are).

1.3 Outline of paper

In §2 we propose a method for adding universal and existential quantification over propositions to first-order logic, as a way of providing similar facilities to those described above, and we show that this extension is conservative. We then consider how to recover witnesses for second-order existential quantifiers, and show that there are significant differences between intuitionistic and classical theories: To recover witnesses in intuitionistic theories we have to restrict applications of rules for disjunction slightly. We prove that this restriction is necessary, and discuss how it relates to restrictions in Isabelle. After, we show how this restriction can be lifted for classical theories. In §3, we give examples of applications, and in §4 we present our conclusions.

2 Quantifiers over formulae, and their theory

2.1 LJ_2

Definition 1 (The theory LJ**)** *We take the theory LJ to be a standard presentation of intuitionistic first-order logic as it can be found in, e.g., Gentzen [5] or Girard [6]: the class of formulae $\mathcal{L}_{LJ}$ is defined as*

$$F ::= A \mid \bot \mid F \wedge F \mid F \vee F \mid F \to F \mid \forall v F \mid \exists v F$$

where A ranges over atomic formulae and v variables. We take $\neg A$ and $A \leftrightarrow B$ as abbreviations for $A \to \bot$ and $(A \to B) \wedge (B \to A)$. Sequents *in LJ are of the form $\Gamma \vdash G$ where Γ is a set of formulae and G is a formula. All* basic sequents *of the form $\Gamma, A \vdash A$ are provable. There is a pair of rules for each connective, to introduce it on the right or the left, that one associates with the sequent calculus, along with the cut rule*

$$\frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\,\mathsf{cut}. \tag{7}$$

The reader should be familiar with the standard proof of the cut elimination theorem for LJ. In particular we need induction on the lexicographically ordered pair $\langle R, S\rangle$, where R is the logical complexity of the cut formula, measured as the number of logical connectives, and S is the depth of the proof tree.

Definition 2 (The language $\mathcal{L}_{LJ_2}$**)** *The language $\mathcal{L}_{LJ_2}$ is the smallest extension of the language $\mathcal{L}_{LJ}$ closed under the following:*

1. *We define a new class of* predicate *variables, α, of all arities, where if α has arity n, then $\alpha(t_1, \ldots, t_n) \in \mathcal{L}_{LJ_2}$ (we call the other variables* first-order *variables to distinguish them from the predicate variables).*

2. *If $A \in \mathcal{L}_{LJ_2}$, and α is a predicate variable, then $\forall_2 \alpha A \in \mathcal{L}_{LJ_2}$ and $\exists_2 \alpha A \in \mathcal{L}_{LJ_2}$.*

3. *If $A \in \mathcal{L}_{LJ_2}$* and does not contain $\forall_2$ or $\exists_2$, *then $\lambda x_1, \ldots, x_n A$, where $x_1, \ldots, x_n$ includes all the free variables of A, is a predicate of $\mathcal{L}_{LJ_2}$ of arity n, and called an* abstracted formula. *When an abstracted formula is substituted into a predicate variable of the appropriate arity, the resulting redex is reduced; e.g. the term $(\ldots \alpha(t_1, \ldots, t_n) \ldots)[\lambda x_1, \ldots, x_n A/\alpha]$ reduces to the term $\ldots A[t_1/x_1, \ldots, t_n/x_n] \ldots$*

Definition 3 (The theory LJ_2**)** *The logic LJ_2 has the language $\mathcal{L}_{LJ_2}$. The rules include the rules for LJ, extended to the $\mathcal{L}_{LJ_2}$ syntax. We also add* *rules for the second-order quantifiers* $\forall_2$ *and* $\exists_2$*, as follows:*

$$\frac{\Gamma, A[p/\alpha] \vdash B}{\Gamma, \forall_2 \alpha A \vdash B}\ \forall_2\text{-l} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \forall_2 \alpha A}\ \forall_2\text{-r}$$
$$\frac{\Gamma, A \vdash B}{\Gamma, \exists_2 \alpha A \vdash B}\ \exists_2\text{-l} \qquad \frac{\Gamma \vdash A[p/\alpha]}{\Gamma \vdash \exists_2 \alpha A}\ \exists_2\text{-r} \tag{8}$$

(where p *is an abstracted formula, and the standard side conditions on* $\forall_2$-r *and* $\exists_2$-l *that* α *not appear free in* Γ, B *hold).*

2.2 Cut-elimination and Conservativity

We now explore the properties of the ramified second-order logic we have defined. We begin by defining a special category of cuts, which we call 'second-order', and prove a cut elimination theorem for it.

Definition 4 (Second-order cuts) *In applications of the cut rule, (7), if* A *is a formula containing either* $\forall_2$ *or* $\exists_2$*, then the cut is called* second-order.

We can now prove the following:

Proposition 5 *If a sequent* $\Gamma \vdash A$ *is provable in* LJ_2 *then it is provable without second-order cuts.*

Proof: The proof is by induction on derivations, and similar to ordinary cut elimination, except that we treat first-order cuts like any other rule. We use a lexicographic ordering on the triple $\langle Q, R, S \rangle$ where R and S are, like in the usual proof, the complexity of the cut formula and the number of steps in the derivation, and Q is the number of second-order quantifiers in the cut formula.

Since most of the cases for the induction are the same as for ordinary cut elimination, we assume familiarity with them, and look only at some of the new ones. Consider $\exists_2$; i.e. where the proof of the cut formula is a proof with $\exists_2$-r as the root rule application on the left:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\ \Pi_1\\ \Gamma \vdash A[p/\alpha]\end{array}}{\Gamma \vdash \exists_2 \alpha A}\ \exists_2\text{-r} \qquad \dfrac{\begin{array}{c}\vdots\ \Pi_2\\ \Gamma', A \vdash B\end{array}}{\Gamma', \exists_2 \alpha A \vdash B}\ \exists_2\text{-l}}{\Gamma, \Gamma' \vdash B}\ \text{cut}$$

which is restructured as

$$\dfrac{\begin{array}{c}\vdots\ \Pi_1\\ \Gamma \vdash A[p/\alpha]\end{array} \qquad \begin{array}{c}\vdots\ \Pi_2[p/\alpha]\\ \Gamma', A[p/\alpha] \vdash B\end{array}}{\Gamma, \Gamma' \vdash B}\ \text{cut}$$

where Π_1 and Π_2 are derivations free of second-order cuts (and $\Pi_2[p/\alpha]$ denotes the derivation that results from substituting through the abstracted formula p for the propositional variable α, which is allowed by the theorem on constants). Notice that $A[p/\alpha]$ has a smaller Q measure, since a second-order quantifier has been removed and, by the restrictions in definition 2.3, p cannot have introduced any new ones. By the hypothesis it is now possible to eliminate the second-order cut from the derivation. The case for $\forall_2$ is very similar. The other extra case is when one of the sub-derivations of a second-order cut (cut_2) is a first-order cut (cut_1):

$$\dfrac{\dfrac{\dfrac{\vdots\,\Pi_1}{\Gamma \vdash A} \quad \dfrac{\vdots\,\Pi_2}{\Gamma', A \vdash B}}{\Gamma, \Gamma' \vdash B}\,\mathsf{cut}_1 \qquad \dfrac{\vdots\,\Pi_3}{\Gamma'', B \vdash C}}{\Gamma, \Gamma', \Gamma'' \vdash C}\,\mathsf{cut}_2$$

which is restructured as

$$\dfrac{\dfrac{\vdots\,\Pi_1}{\Gamma \vdash A} \qquad \dfrac{\dfrac{\vdots\,\Pi_2}{\Gamma', A \vdash B} \quad \dfrac{\vdots\,\Pi_3}{\Gamma'', B \vdash C}}{\Gamma', \Gamma'', A \vdash C}\,\mathsf{cut}_2}{\Gamma, \Gamma', \Gamma'' \vdash C}\,\mathsf{cut}_1$$

and, since the S measure of the subderivation ending with the second-order cut is now smaller, it can, by appeal to the hypothesis, be restructured to be cut free; when the cut is on the right hand side the transformation is similar. And the theorem follows. ∎

Readers familiar with cut-elimination can see that all cuts can be eliminated; but, in general, we cannot do this for an arbitrary theory that we might define as an axiomatic extension of LJ_2. We discuss this shortly (corollary 8).

We now point out that second-order cut free proofs enjoy a property similar to the subterm property for cut free proofs:

Remark 6 *If a proof is free of second-order cuts then application of any of the rules in (8) would result in a second-order quantifier appearing in the root sequent. This follows since the only way such quantifiers, after having been introduced by a rule application, could disappear before the root sequent, is by a second-order cut, and there are no such cuts in the proof.*

Now we can give a conservativity result, where conservativity is defined as follows:

Definition 7 (Conservativity) *A theory T' with language $\mathcal{L}'$ is conservative over a theory T with language $\mathcal{L}$ if any formula F of $\mathcal{L} \cap \mathcal{L}'$ provable in T' is provable in T.*

We do not want just to show that LJ_2 is conservative over LJ, but rather something more general and useful: that LJ_2 extended to some theory like arithmetic is conservative over the first-order version. Thus define $LJ_2 + Ax$, where Ax is a set of formulae, to be the theory LJ_2 extended with all sequents of the form $\vdash A$, where A is in Ax; and define Ax^* to be the set of formulae generated from Ax by every possible predicate-variable-free instantiation of predicate variables in Ax. Then we show that

Corollary 8 *For any set of formulae Ax not containing second-order quantifiers, the theory $LJ_2 + Ax$ is conservative over $LJ + Ax^*$.*

Proof: This follows because the transformations used earlier in the procedure for eliminating second-order cuts are still allowed, so we can transform a proof in $LJ_2 + Ax$ of a formula F in the language $\mathcal{L}_{LJ}$ into a proof free of second-order cuts, and thus, by remark 6, a proof in which none of (8) are used. Predicate variables might still occur in the proof, having been introduced by base sequents either from LJ_2 or Ax. In either case, choose an instantiation of these free of propositional variables. The result of this is a derivation in $LJ + Ax^*$. ■

Remark 9 *By the corollary we can immediately implement, for instance, any classical first-order theory in LJ_2 by taking Ax to be of the form $Ax \cup \{\forall x_1, \cdots \forall x_n(\alpha(x_1, \ldots, x_n) \vee \neg\alpha(x_1, \ldots, x_n) \mid n \text{ a natural number}\}$ where α is a n-ary predicate variable.*

We now have machinery sufficient to allow us to do what Goguen, and Boyer *et al.* do: for an arbitrary first-order theory we can formalize and prove the universal closure of schematic first-order formulae like (2). Our conservativity result guarantees that their use does not result in any more first-order theorems becoming provable.

2.3 Realising existential predicate variables

The logic we have developed provides more than is needed for formalizing and using first-order schematic proof rules: if this was all that we wanted, we could delete the second-order existential quantifiers and associated rules from LJ_2. However we want to work with formulae like (6). To this end, ideally we would like a theorem as follows:

Hypothesis 10 *Given a proof of a formula $\exists_2 \alpha A$ in LJ_2 we can transform this into a proof of $A[p/\alpha]$ for some abstracted formula p.*

Unfortunately, this is false. The natural way to construct a witness for α is by structural recursion on the proof, but we immediately encounter a problem with this, when we try to deal with the rule for introducing $\vee$ on the left:

$$\frac{\Gamma, A \vdash \exists_2 \alpha P(\alpha) \quad \Delta, B \vdash \exists_2 \alpha P(\alpha)}{\Gamma, \Delta, A \vee B \vdash \exists_2 \alpha P(\alpha)} \vee\text{-l} \tag{9}$$

There is no obvious way to construct a witness for the α in the goal, given witnesses for the two instances of α in the subgoals. In fact, this generally cannot be done.

Proposition 11 *It is not possible to find a witness for the outside second-order existential quantifier in a theorem in an arbitrary extension of LJ_2.*

Proof: We give a simple first-order theory T, a theorem $\exists_2 \alpha P(\alpha)$ of $LJ_2 + T$ where P contains no second-order quantifiers, and a (Kripke) model of $LJ + T$ where no instance of $P(\alpha)$ is true. From this it follows that there is no instantiation of α for which $P(\alpha)$ is provable.

First we give the facts we need about Kripke models of intuitionistic logic. Our account comes directly from [16, §5.3], and we only remind the reader what is critical to follow our example. A model of intuitionistic predicate logic is a quadruple $\mathfrak{M} = \langle \mathrm{K}, \Sigma, \mathrm{C}, \mathrm{D} \rangle$ that meets a number of conditions. Relevant for our example is that K is a partially ordered set of worlds and Σ is a function from worlds to sets of atomic sentences that states which basic facts hold at each world (we do not need C and D here; we include them simply to be consistent with [16]). Facts are cumulative in the sense that for $k, l \in \mathrm{K}$, if $k \leq l$ then $\Sigma(k) \subseteq \Sigma(l)$. We write $k \Vdash A$, for the atomic fact A when $A \in \Sigma(k)$; this may be extended to a unique relation over all formula (we consider only propositional connectives here) as follows:

- $k \Vdash A \wedge B$ if $k \Vdash A$ and $a \Vdash B$.
- $k \Vdash A \vee B$ if $k \Vdash A$ or $k \Vdash B$.
- $k \Vdash A \rightarrow B$ if for all $l \geq k$, $l \Vdash A$ implies $l \Vdash B$.

The completeness property we need is that $\Gamma \vdash A$ iff for all models $\mathfrak{M}$, and worlds k in $\mathfrak{M}$, if $k \Vdash \Gamma$ then $k \Vdash A$ [16, §5.3.10].

Take A, B, C, D to be propositional constants and consider the theory

$$T \equiv \{A \vee B\}$$

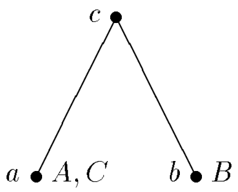
and the formula

$$P(\alpha) \equiv (A \wedge (C \leftrightarrow \alpha)) \vee (B \wedge (D \leftrightarrow \alpha)).$$

We can prove that $\exists_2 \alpha P(\alpha)$ in $LJ_2 + T$:

$$\dfrac{\vdash A \vee B \qquad \dfrac{\dfrac{\overset{\vdots}{A \vdash P(C)}}{A \vdash \exists_2 \alpha P(\alpha)}\,\exists_2\text{-r} \qquad \dfrac{\overset{\vdots}{B \vdash P(D)}}{B \vdash \exists_2 \alpha P(\alpha)}\,\exists_2\text{-r}}{A \vee B \vdash \exists_2 \alpha P(\alpha)}\,\vee\text{-l}}{\vdash \exists_2 \alpha P(\alpha)}\,\text{cut}$$

By the completeness of Kripke models of intuitionistic logic, if there is an α such that $A \vee B \vdash P(\alpha)$ then in every model $\mathfrak{M}$ and world k, if $k \Vdash A \vee B$ then $k \Vdash P(\alpha)$. But consider a Kripke model consisting of three worlds $a, b \leq c$, arranged as follows:



Here only the explicitly stated atomic propositions, or those explicitly forced in some earlier world, are forced, thus $A \vee B$ is forced at each world, since either A or B is forced at each world. Now, suppose $P(\alpha)$ is provable. Then it is forced everywhere, in particular at a. And there, since $a \Vdash A$ and $a \nVdash B$, we must have that $a \Vdash C \leftrightarrow \alpha$, and thus that $c \Vdash C \leftrightarrow \alpha$. Further, since $c \Vdash C$ we must have $c \Vdash \alpha$. Now consider b. Since $b \Vdash P(\alpha)$ and $b \Vdash B$, but $b \nVdash A$, we must have that $b \Vdash D \leftrightarrow \alpha$. But this implies that $c \Vdash D \leftrightarrow \alpha$ and, since $c \Vdash \alpha$ also $c \Vdash D$. But this is a contradiction since D is not forced at any world. We conclude that $P(\alpha)$ is not provable. ■

Compare what happens in Isabelle to (9); then the rule is

$$\frac{\Gamma, A \vdash P(\alpha) \quad \Delta, B \vdash P(\alpha)}{\Gamma, \Delta, A \vee B \vdash P(\alpha)} \tag{10}$$

where α is a single metavariable, not two existentially quantified object predicate variables. This means that we are able to instantiate α as we like, but this instantiation takes place over the entire proof, not just in a sequent at some particular leaf of the tree; i.e. Isabelle will ensure that the occurrences of α in the proof are instantiated identically. This sharing across subproofs introduces 'non-local' effects into the derivation: by doing something at one subgoal that instantiates α in some way, another subgoal also containing α can become suddenly unprovable. We can, in fact, see the problem with combining the two instances of α in (9) as simply the result of not insisting, as Isabelle does, that the two witnesses are the same.

We have shown that, in general, we cannot recover a witness for an existentially quantified predicate variable. In §2.5 we will show that if we restrict ourselves to classical theories, this result no longer holds; in fact we can then recover witnesses in some circumstances beyond what Isabelle allows. On the other hand, in Isabelle we can work with metavariables in, e.g., intuitionistic logics, and further, the terms we synthesise using metavariables are not the same as the terms generated by the methods discussed in §2.5. So, before looking at what happens in classical logic, we investigate the general case that corresponds more closely to metavariables as we find them in, e.g. Isabelle.

2.4 Realisability for intuitionistic theories

Since the rule $\vee$-I (i.e. (9)) is the source of our problems in trying to develop a proof-theoretic account of realizability for metavariables, we now consider what sort of restrictions we have to place on its use in derivations in LJ_2, in order to avoid those problems.

It turns out that a small side condition is what we need:

Definition 12 (The theory LJ_2^R) *The language of LJ_2 is:*

$$F_2 ::= A \mid \bot \mid F_2 \wedge F_2 \mid F_2 \vee F_2 \mid F_2 \rightarrow F_2 \mid \forall v F_2 \mid \exists v F_2 \mid \forall_2 v F_2 \mid \exists_2 v F_2.$$

where A is atomic predicates. We define a subclass of this as:

$$F_2^R ::= A \mid \bot \mid F_2^R \wedge F_2^R \mid F_2^R \vee F_2^R \mid F_2 \rightarrow F_2^R \mid \forall v F_2^R \mid \exists v F_2^R \mid \forall_2 v F_2^R$$

(i.e. we allow second-order existential quantifiers only inside the scope of the antecedent of some implication). Then we place the restriction on $\vee$-I that it can only be used when the consequent of the sequent (i.e. $P(\alpha)$ in (9)) is in F_2^R. Call the theory with this restriction LJ_2^R.

Because LJ_2^R is a subtheory of LJ_2 we can eliminate cuts from derivations in it just like in LJ_2, so the conservativity results for LJ_2 still hold (the restriction does not prevent any derivations in the subtheory LJ, since it rules out only certain formulae with quantifiers over propositions). However the transformed proof may no longer be a proof LJ_2^R but only in LJ_2; i.e. there may be occurences of $\vee$-I where the consequent is of the form $\exists_2 \alpha A$. Consider the example

$$\dfrac{\dfrac{A, \Gamma \vdash G \quad B, \Delta \vdash G}{A \vee B, \Gamma, \Delta \vdash G}\,\vee\text{-I} \qquad \Lambda, G \vdash \exists_2 \alpha C}{A \vee B, \Gamma, \Delta, \Lambda \vdash \exists_2 \alpha C}\,\text{cut}$$

(where G is some formula in F_2^R) the proof gets permuted to

$$\dfrac{\dfrac{A, \Gamma \vdash G \quad \Lambda, G \vdash \exists_2 \alpha C}{A, \Gamma, \Lambda \vdash \exists_2 \alpha C}\,\text{cut} \qquad \dfrac{B, \Delta \vdash G \quad \Lambda, G \vdash \exists_2 \alpha C}{B, \Delta, \Lambda \vdash \exists_2 \alpha C}\,\text{cut}}{A \vee B, \Gamma, \Delta, \Lambda \vdash \exists_2 \alpha C}\,\vee\text{-I}$$

which is not permitted in LJ_2^R (because now the consequent is existentially quantified). However we can still recover a witness if we are a bit more careful about the cuts that we eliminate. Thus we have the theorem

Theorem 13 *Given a proof of $\vdash \exists_2 \alpha A$ in LJ_2^R, we can find some abstracted formula p and a proof of $\vdash A[p/\alpha]$ in LJ_2^R.*

Proof: Given any proof of $\vdash \exists_2 \alpha A$ in LJ_2^R, there is a chain of rule applications extending from the root of the proof all of which have $\exists_2 \alpha A$ as a consequent—call this the path. There are no branches on the path, because no applications of the rule $\vee$-l are permitted. The path may, however, run through first or second-order cuts which introduce new assumptions. We show that we can remove enough second-order cuts from the path so that it is possible to recover a witness for the existential while remaining in LJ_2^R.

We call second-order formulae that have been cut into the path, but which are not in the class F_2^R 'safe' because we can safely apply the cut-elimination transformations to such cuts—there is no risk of such a transformation introducing an $\vee$-l onto the path (since by the definition of LJ_2^R there cannot be an $\vee$-l above the cut formula which could be introduced), which would remove the derivation from LJ_2^R. Thus we can remove all safe second-order cuts from the path.

Consider the path now. There are two cases to consider: either it ends at an application of $\exists_2$-r or at a basic sequent. Since the consequent of the sequent in the path is safe, if the path is to end in a basic sequent then somehow a safe formula has to appear in the hypothesis list. This cannot get there directly, by a cut, since there are now no safe cuts in the path, however it might be possible to manipulate some unsafe formula that has been cut-in, to produce a safe one. We can show that this cannot happen, by induction on the structure of F_2^R. The only interesting case is when the unsafe formula is of the form $A \to B$ where A is possibly safe; the only way we can manipulate such a term is by $\to$-l, which leaves only the unsafe B on the path, the possibly safe A gets moved off as the goal of a sub-derivation. Thus there can be no safe formulae on the path, so the path cannot end in a basic sequent.

In the first case, we can simply delete the rule, and the result will still be a proof, but of $\vdash A[p/\alpha]$ for some p. Notice that because of the way we have defined abstracted formulae, no new variable can suddenly appear on the path as a result of substituting p through for α, to make some application of $\exists$-l suddenly invalid, and we have already ensured that there are no instances of $\exists_2$-l on the path since there are no safe second-order cuts on it. ■

In the process of finding the witness for a second-order quantifier in LJ_2^R we do not eliminate all the second-order cuts. However the relationship between what happens in Isabelle and what is, essentially, happening here, is made clear by the following observation.

Remark 14 *If we eliminate all the second-order cuts, what we end up with is a proof in LJ_2 where if $\vee$-I occurs with a consequence that is not safe, then the witnesses that we recover for the two subproofs are identical (so there is no problem with combining them to produce a witness for the full proof).*

This is, as we pointed out before, the condition that is enforced in Isabelle, where unification is used to fill out metavariables occurring in different branches of a proof identically.

2.5 Realizability restricted to classical theories

So far we have been thinking of general extensions of intuitionistic logic, including classical logic. This has meant that we have had to place some side conditions on the proof rules. But, observe that if we restrict ourselves to classical theories (c.f., remark 9), then we can remove these conditions: we have that

Proposition 15 *Given a proof of $\exists_2 \alpha A$ in $Cl_2 + T$ we can transform this into a proof in $Cl_2 + T$ of $A[p/\alpha]$ for some abstracted formula p.*

Proof: The point of the restrictions in LJ_2^R is to ensure that we are able to recover witnesses for existentially quantified predicate variables, and we cannot do this in the case of an application of $\vee$-I: the problem is, in the circumstance of (9), given witnesses α_A and α_B for α in $\Gamma, A \vdash \exists_2 \alpha P(\alpha)$ and $\Delta, B \vdash \exists_2 \alpha P(\alpha)$, to find a formula $F(\alpha_A, \alpha_B, A, B)$ that is logically equivalent to exactly one of α_A or α_B, depending on whether A is true or false. But in a classical theory we can use the term

$$(A \wedge \alpha_A) \vee (\neg A \wedge \alpha_B) \tag{11}$$

since we know, *a priori*, that $A \vee \neg A$. ∎

Aside from the fact that this approach fails for intuitionistic theories, it introduces additional complexity not present when manipulating metavariables directly in schematic proofs. In particular, the witnessing formula is more complex when $\vee$-I is applied to a sequent whose goal is quantified with $\exists_2$. This additional complexity is because a wider class of proofs is possible here than in Isabelle: as we pointed out earlier, there are proofs which use $\vee$-I duplicate a formula containing a metavariable onto two branches, so that the metavariable is instantiated identically in each branch. Here we are allowed more freedom and, as a result, require a more complex realizer. Note however that when both branches are instantiated identically we can simplify (11) to α_A, which is identical to what we get in Isabelle.

Finally, we point out that the theories that we have described here, with their associated results, are essentially fragments of the ramified higher-order logics that Prawitz discusses [14, V.§2] (based on natural deduction, rather than the sequent calculus used here). However, Prawitz's normalization results are for the pure form of his logic and we are primarily interested in extensions with arbitrary first-order theories, where such normalization is not possible. Further, we are concerned with finding witnesses for second-order existential quantifiers in such extensions (which we get by the restriction we place on abstracted formulae in definition 2.3).

3 Applications

With the machinery that we have developed, we reexamine the kinds of extensibility discussed in §1.

3.1 Universal quantification

Consider a more complex example than (1): if we work in a theory like arithmetic, we might like to use some other induction schema than the standard structural induction provided in Peano Arithmetic (PA); e.g. course-of-values induction

$$\forall x(\forall y(y < x \to A[y/x]) \to A) \to \forall x A.$$

In a system like Isabelle we can prove this schematic version directly, with A universally quantified at the metalevel.

Now, take AR_2 as LJ_2 extended with the axioms of PA, with a single schematic version of the induction axiom, instead of the infinite set of instances. By corollary 8 this is conservative over the first-order axiomatization of PA. We can easily show that:

Proposition 16 *In the theory AR_2:*

$$\vdash \forall_2 \alpha(\forall x(\forall y(y < x \to \alpha(y)) \to \alpha(x)) \to \forall x \alpha(x)).$$

Proof: We follow the standard textbook proof and assume an arbitrary α and the antecedent of $\to$. After, we generalize to $\forall z \forall t(t < z \to \alpha(t))$ which we prove using structural induction on z: the base case, $z = 0$ follows vacuously since $t < z$ is false, while, in the step case, we have to show that $\forall t(t < s(z) \to \alpha(t))$ under the hypothesis $\forall t(t < z \to \alpha(t))$. For $t < z$ we use the hypothesis and, for $t = z$, the antecedent of the original goal follows again by the hypothesis. This argument can be formalized in AR_2, since the induction formula contains no second-order quantifiers. ∎

Now we have a single theorem, which we can store as a lemma, and use to generate instances of course-of-values induction as necessary. But this is not quite enough yet: we have proven the course-of-values schema only for formulae α with exactly one parameter. If we need an instance where α has additional parameters, we would seem to need a new schema with additional parameters. However we can instead make use of the simple metatheorem about our calculus.

Lemma 17 *If a theorem F contains a quantified predicate variable α of arity n, then if we uniformly substitute $\alpha(t_1, \ldots, t_n)$ through with $\alpha(t_1, \ldots, t_n, t_{n+1})$, where t_{n+1} is an arbitrary term (avoiding capture), then the new formula is also a theorem.*

Proof: The proof is by structural induction on proofs. The base case is trivial, and the important subcases of the step case are the rules (8): it is easy to see how the applications of these must be modified in the proof so that the new formula is proven; for instance, for the $\forall_2$-I rule we just have to add a new lambda abstracted variable to the arguments of the abstracted predicate p which will pick up the new term t_{n+1}. ■

For the reader familiar with [11], this metatheorem directly corresponds to how rules are lifted over context parameters in systems like Isabelle.

3.2 Existential quantification

Consider a simple synthesis example: converting a propositional formula to its conjunctive normal form (cnf) in classical logic. We look at how this works in a system like Isabelle, and in our logic.

We can give rules for converting formulae to cnf as a set of rewrite rules.

$$\begin{array}{rcl} \neg\neg\alpha & \rightsquigarrow & \alpha \\ \neg(\alpha \vee \beta) & \rightsquigarrow & \neg\alpha \wedge \neg\beta \\ \alpha \vee (\gamma \wedge \delta) & \rightsquigarrow & (\alpha \vee \gamma) \wedge (\alpha \vee \delta) \end{array}$$

We can formulate these in Isabelle as derived rules, which we can use to synthesize predicates; e.g. the second can be given as

$$\frac{\neg\alpha \leftrightarrow \gamma \quad \neg\beta \leftrightarrow \delta}{\neg(\alpha \vee \beta) \leftrightarrow \gamma \wedge \delta}\,\mathsf{rw}\,.$$

Consider how we would use these to transform a formula like $\neg(p_1 \vee (p_2 \vee p_3))$ into cnf in Isabelle. We begin with the goal

$$\neg(p_1 \vee (p_2 \vee p_3)) \leftrightarrow \alpha \tag{12}$$

and apply rules until we are left with atomic propositions, at which point we use the fact that $(\alpha)(\vdash \alpha \leftrightarrow \alpha)$ to instantiate remaining metavariables. An Isabelle proof (annotated with the substitutions generated by unification) looks like

$$\frac{\dfrac{\neg p_1 \leftrightarrow \neg p_1}{\neg p_1 \leftrightarrow \gamma}\,\{\neg p_1/\gamma\} \qquad \dfrac{\dfrac{\neg p_2 \leftrightarrow \neg p_2}{\neg p_2 \leftrightarrow \gamma'}\,\{\neg p_2/\gamma'\} \quad \dfrac{\neg p_3 \leftrightarrow \neg p_3}{\neg p_3 \leftrightarrow \delta'}\,\{\neg p_3/\delta'\}}{\neg(p_2 \vee p_3) \leftrightarrow \delta}\,\mathsf{rw}\,\{\gamma' \wedge \delta'/\delta\}}{\neg(p_1 \vee (p_2 \vee p_3)) \leftrightarrow \alpha}\,\mathsf{rw}\,\{\gamma \wedge \delta/\alpha\} \tag{13}$$

We can see how the steps in the proof incrementally generate the substitution $\neg p_1 \wedge (\neg p_2 \wedge \neg p_3)$ for α, resulting in the proof we want.

We can carry out this sort of synthesis proof in Cl_2 as follows. First, instead of deriving the rewrite rules as proof rules, we prove them as theorems where existentially quantified variables stand in for the missing formulae that need to be constructed; e.g. the rewrite rule we gave for Isabelle as rw, corresponds to the Cl_2 theorem:

$$\forall_2 \alpha\, \beta (\exists_2 \gamma (\neg \alpha \leftrightarrow \gamma) \to \exists_2 \delta (\neg \beta \leftrightarrow \delta) \to \exists_2 \phi (\neg(\alpha \vee \beta) \leftrightarrow \phi)). \tag{14}$$

This is easy to do under the LJ_2^R restriction since for arbitrary α, β, and given γ and δ satisfying the assumptions, ϕ can be instantiated with $\gamma \wedge \delta$. Note that the way we prove this lemma is important. We could have also proved it by instantiating ϕ with $\neg(\alpha \vee \beta)$, ignoring γ and δ. But this would not result in a lemma useful for CNF normalization. This kind of 'intensional' concern when proving theorems is similar to what we have to pay attention to in constructive logics for program synthesis: if we synthesize a program for some specification we do not want just any program, but often have a particular, efficient, program in mind.

In Isabelle, rewrite rules are applied using unification, which we do not have here. Instead we use a simple tactic to apply a set of theorems like (14), which cuts them in and then backchains through them; i.e. the variables α and β in 14 are instantiated by matching[2] the consequent against the current goal and then $\to$-l is repeatedly applied, to leave the antecedents as the remaining subgoals. Call this tactic rt.

Now take the existential closure of (12), $\exists_2 \alpha (\neg(p_1 \vee (p_2 \vee p_3)) \leftrightarrow \alpha)$, as the goal. The proof is built by repeated applications of rt to reduce the subgoals to equivalences between atomic formulae and metavariables and then instantiating the existential variables with the appropriate atomic predicates:

$$\dfrac{\dfrac{\vdash \neg p_1 \leftrightarrow \neg p_1}{\vdash \exists_2 \gamma \neg p_1 \leftrightarrow \gamma}\,\exists_2\text{-}\mathsf{r} \qquad \dfrac{\dfrac{\vdash \neg p_2 \leftrightarrow \neg p_2}{\vdash \exists_2 \gamma' (\neg p_2 \leftrightarrow \gamma')}\,\exists_2\text{-}\mathsf{r} \qquad \dfrac{\neg p_3 \leftrightarrow \neg p_3}{\vdash \exists_2 \delta' (\neg p_3 \leftrightarrow \delta')}\,\exists_2\text{-}\mathsf{r}}{\vdash \exists_2 \delta (\neg(p_2 \vee p_3) \leftrightarrow \delta)}\,\mathsf{rt}}{\vdash \exists_2 \alpha (\neg(p_1 \vee (p_2 \vee p_3)) \leftrightarrow \alpha)}\,\mathsf{rt}$$

Remember that the steps labeled rt are not atomic rule applications, but derivations using second-order cuts to introduce instances of (14). If we 'unwind' this proof by eliminating the second-order cuts using the algorithm in the proof of theorem 13, and label the transformed proof with the substitutions that we generate as we do so, we see that the decorated proof is

[2] In this case and many others, although there are second-order variables, matching and unification are restricted to so called 'higher order pattern unification' [10] which like first-order unification is linear time computable and has a unique most general unifier.

identical to (13), except for the added application of $\exists_2$-r at the bottom. Also, even though we are working in the theory Cl_2 our proof is compatible with theorem 13; we did not need to resort to proposition 15.

4 Conclusions

We have proposed a way to extend intuitionistic or classical first-order theories conservatively with quantification over formulae as a way to provide what we normally regard as metatheoretic facilities, while introducing as few changes as possible into the logic. In part, we have tried to emulate what is available in a system like Isabelle. Our second-order universal quantifiers can be seen as generalizing the kinds of schematic lemma facilities that Goguen, and Boyer et al. propose, and, together with existential quantification, as reproducing some of the metatheoretic facilities available in a framework proof development system. Moreover our proof-theoretic proofs can be contrasted with Goguen's semantic proofs of the conservativity of second-order quantifiers [7, appendix]. We are sympathetic to the relative clarity and brevity of semantic proofs in general, but our proof theoretic analysis does not commit us to a particular semantics and allows us, in the end, to remove systematically all traces of the new machinery that we have introduced. Our second-order existential quantifiers are similar, in some respects, to what we find in a constructive higher-order logic. However, we only require restricted kinds of proof normalization; we do not require normalization of the first-order component (indeed, we want to avoid this) nor, for example, that the theories satisfy the disjunction property. Finally, unlike in a framework system, our extensions are part of the logic itself, rather than the result of a two-level system with a formal metalogic.

Acknowledgments

We thank Alan Smaill, Nax Mendler, and the anonymous referees for their careful reading of drafts and critical comments. This research was partly supported by the German Ministry for Research and Technology under grant ITS 9102. Responsibility for the contents of this paper lies with the authors.

References

[1] A. Avron, F. Honsell, I. Mason, and R. Pollack. Using typed lambda calculus to implement formal systems on a machine. *J. Automated Reasoning*, 9:309–352, 1992.

[2] D. Basin, A. Bundy, I. Kraan, and S. Matthews. A framework for program development based on schematic proof. In *Proc. 7th Int.*

Workshop on Software Specification and Design, California, 1993. IEEE press.

[3] R. Boyer, D. M. Goldschlag, M. Kaufmann, and J. S. Moore. Functional instantiation in first-order logic. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation. Papers in honour of John McCarthy*. Academic Press, New York, 1991.

[4] T. Clement. Using metavariables in natural deduction proofs. In C. B. Jones, R. C. Shaw, and T. Denvir, editors, *5th Refinement Workshop*, pages 255–271, London, 1992. Springer.

[5] G. Gentzen. Untersuchen über das logische Schliessen. *Math. Zeitschrift*, 39:179–210, 405–431, 1934.

[6] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, Cambridge, 1989.

[7] J. Goguen. Higher-order functions considered unnecessary for higher-order programming. In D. A. Turner, editor, *Research Topics in Functional Programming*. Addison-Wesley, Massachusetts, 1990.

[8] J. Hesketh, A. Bundy, and A. Smaill. Using middle-out reasoning to transform naive programs into tail recursive ones. In D. Kapur, editor, *CADE–11*, volume 607 of *LNCS*. Springer, 1992.

[9] G. Huet and G. Plotkin, editors. *Logical Frameworks*. Cambridge University Press, Cambridge, 1991.

[10] T. Nipkow. Higher-order critical pairs. In *Symposium on Logic in Computer Science*, pages 342–349. Computer Society Press of the IEEE, 1991.

[11] L. C. Paulson. The foundation of a generic theorem prover. *J. Automated Reasoning*, 5(3):363–397, 1989.

[12] L. C. Paulson. *Isabelle: A generic theorem prover*, volume 828 of *LNCS*. Springer, Berlin, 1994.

[13] F. Pfenning. Logic programming in the LF logical framework. In Huet and Plotkin [9], pages 149–181.

[14] D. Prawitz. *Natural Deduction; A proof theoretic study*. Almqvist and Wiksell, Stockholm, 1965.

[15] D. J. Pym and L. Wallen. Proof-search in the $\lambda\Pi$-calculus. In Huet and Plotkin [9], pages 309–340.

[16] D. van Dalen. *Logic and Structure*. Springer, Berlin, third edition, 1994.