Unskolemizing

by

W. W. Bledsoe and A. Michael Ballantyne

July 1978 ATP-41A

(Preliminary Version)

This work was supported by the National Science Foundation Grant MSC77-20701.

Unskolemizing

by

W. W. Bledsoe and A. Michael Ballantyne

ABSTRACT. We give here a procedure for obtaining an equivalent, completely quantified, form for a given skolemized formula, and show some examples of its use. We also discuss briefly the automatic generation of counterexamples, and how the unskolemization process is related to that.

1. Introduction

Most automatic theorem provers, including our [1-3], remove all quantifiers from the theorem before the actual proving takes place. This is done by "skolemization" (See for example, App. 1 of [1].)

Sometimes it is desirable to recover the quantification by reversing this process of skolemization. For example, when a counterexample is being sought to a particular subgoal in a proof, it is important to obtain the completely quantified form of that subgoal so that it can be negated and proved.

At first glance this would seem a simple task, which can be achieved by remembering what was done during skolemization and reversing it, and that indeed is the case when the formula being unskolemized is the whole original theorem. However, if we are dealing with a fragment of it, or a subgoal gotten from it by using a lemma, then it is not obvious _what_ the proper quantification is, let alone how to obtain it.

The purpose of this note is to give an algorithm for such "unskolemization" (Section 2), and to give several examples of its use (Section 3). In Section 4 we briefly address the question of automatically obtaining counterexamples to a suspected theorem, and discuss how unskolemization is involved in that, and give some examples.

2. Algorithms

Let A' be a completely quantified formula (WFF), and let A be its skolemized form. We wish to reconstruct A' from A. We wish also to handle cases where A was not obtained directly from such an A' by skolemization, but has been derived from another skolemized expression by instantiating certain of its variables, as might be done by the use of a lemma in the proof of a theorem (See Ex. 3-7, pp. 10-13.)

It is assumed that each skolem expression in A is detectable. (i.e., we can tell which expressions are skolem expressions.) This is accomplished during the skolemization process in our programs by including the letter "S" (for Skolem) as the second letter of the new function name. For example, the expression

$$(\forall x\ \exists y\ \ P(x,y) \rightarrow Q)$$

is skolemized as

$$(P(x,(yS\ x)) \rightarrow Q).$$

Thus the variable y has been replaced by the skolem expression (yS x). At a later time this can be detected as a skolem expression by examining the second letter of yS (by the LISP function EXPLODE). (If other y's are encountered in the formula being skolemized, then uniqueness is assured by using yS2, yS3, etc.) Similarly, a skolem constant, such as "a", is replaced by (AS), etc. Skolem variables are represented as atoms in LISP and are detected that way.

In the following algorithms, which are written in LISP, parenthetical remarks are given by (* ...).

```
(RECONSTRUCT-QUANTIFIERS  (λ(A))
(SETQ  L  (SKO-VARIABLES-EXPRESSIONS  A))
(SETQ  LL (SKO-FRONT  L  NIL))
(SETQ  L' (SUBLISS-NIL  LL  L))
(QUANTIFY  LL  T  NIL (RECON-SKO  A  T  L'))
```

This routine accepts a skolemized expression (WFF) A and returns a fully quantified equivalent expression A'.

It first selects a list L of all the skolem variables and skolem expressions in A, and then collects into LL the first of each of those expressions in L, which are repeated with the same header (e.g., (xS17 y) and (xS17 (AS))), and replaces them in L by NIL. It then uses the routine RECON-SKO to insert quantifiers in A corresponding to the list L, and finally places quantifiers on the "outside" of A corresponding to the list LL. For example if A is

$$[P(X, (YS\ X)) \wedge Q((ZS\ X)) \rightarrow R(0, (ZS\ (CS)))]^{*}$$

then

```
 L = (X  (YS  X)  (ZS  X)  (ZS  (CS))  (CS))
LL = ((ZS  X))
L' = (X  (YS  X) NIL  NIL  (CS))
```

and A' is

$$\forall ZS\ \forall C\ [\forall X\ \exists Y(P(X,Y) \wedge Q((ZS\ X))) \rightarrow R(0, (ZS\ C))].$$

*Actually the program uses a prenex notation

$$(\rightarrow (\wedge\ (P\ X\ (YS\ X))\ (Q\ (ZS\ X)))\ (R\ O\ (ZS\ (CS)))$$

etc.

```
(SUBLISS-NILL (λ(LL L) (COND
[(NULL LL) L]
[(ATOM (CAR L)) (SUBLISS-NIL (CDR LL)L)]
[T (SUBLISS-NIL(CDR LL) (SUBST-CAR NIL (CAAR LL) L))]   )))
```

This replaces by NIL those units in L of the form (XS.. ...) if there is an expression of this form in LL.

```
(SUBST-CAR (λ(X Y A) (COND
[(ATOM A) A]
[(EQ (CAR A) Y) X]
[T(CONS (SUBST-CAR X Y (CAR A))
        (SUBST-CAR X Y (CDR A)) )]   )))
```

This replaces all sub-expressions in A of the form (Y ...) by X. For example if A is (P((XS13 A)) → Q((XS13 Y),Z)) then (SUBST-CAR XS13 XS13 A) returns (P(XS13) → Q(XS13,Z)).

```
(OCCUR-CAR (λ(X L) (COND
[(NULL L) NIL]
[(AND (NOT (ATOM (CAR L))) (EQ X (CAAR L)) T]
[T (OCCUR-CAR X (CDR L))]   )))
```

This determines whether L has any units of the form (X ...) .

```
(QUANTIFY (λ (L P PS A)
   (PROG (QA QS SYM)
(SETQ QA (COND [P 'ALL] [T 'SOME]))
(SETQ QS (COND [P 'SOME] [T 'ALL]))
(SETQ L (SORT-SKO L T))
(RETURN (COND
[(NULL L) A]
[(NULL (CAR L)) (QUANTIFY (CDR L) P PS A)]
[(ATOM (CAR L)) (LIST QS (CAR L) (QUANTIFY (CDR L) P PS A)]
[T (SETQ SYM (CAAR L))
   (LIST QA SYM (QUANTIFY (CDR L) P PS (COND  [PS (SUBST-CAR SYM SYM A)]  [T A])))]
   )))))
```

This accepts a list L of skolem variables and skolem expressions, orders them according to membership and inclusion (see ORDER-SKO), and constructs around A the quantifiers corresponding to the entries in list L.

```
(SORT-SKO (λ(L LT) (COND
[(OR (NULL L) (NULL (CDR L))) L]
[(NULL (CAR L)) (SORT-SKO (CDR L) LT)]
[(AND LT (SETQ L (CONS (CAR L)(SORT-SKO (CDR L) T))) NIL)]
    (*If LT is "on", sort (Cdr L))
[(ATOM (CAR L)) (COND
    [(ATOM (CADR L)) (FLIP)]
    [(MEMBER (CAR L) (CDADR L)) L]
    [T (FLIP)] ) ]
[(ATOM (CADR L)) (COND
    [(MEMBER (CADR L)(CAR L)) (FLIP)]
    [T L]) ]
[(SUBSET (CAR L)(CADR L)) L]
[T (FLIP)]    )))
```

This orders the list L of skolem variables (atoms) and skolem expressions according to membership and inclusion. Note that if A is an atomic formula and F_1 and F_2 are two skolem expressions in A then (cdr F_1) $\subseteq$ (cdr F_2) or vice versa.

```
(FLIP (λ( )
    (CONS (CADR L)
          (SORT-SKO (CONS (CAR L)(CDDR L)) NIL ))   ))
```

This interchanges the first and second members of L and resorts.

3. Examples

Example 1. L = ((f x y z) y z(k x) (ℓ) (g x y z) (j x y) x)

(SORT-SKO L T)

= ((ℓ) x (k x) y (j x y) z (f x y z) (g x y z))

Even though we said in the introduction, p. 1, that skolem functions such as "f" would be atoms where the second letter is an "S", such as fS17, we have suppressed that in some of these examples for brevity of presentation. Thus we have written here k instead of kS··, ℓ instead of ℓS··, g instead of gS··, etc.

Example 2. $\exists x \; \forall y \; (P(y) \wedge Q(x) \rightarrow P(x))$

Skolemized form: $A \equiv P((yS\ x)) \wedge Q(x) \rightarrow P(x))$.

L = ((yS x) x), LL = NIL

SORT-SKO L) = (x (yS x))

(RECONSTRUCT-QUANTIFIERS A) =

$\exists x \; (\exists \; y \;\; P(y) \wedge Q(x) \rightarrow P(x))$

Notice that our reconstructed formula is not exactly equal to the original but it is equivalent to it and in mini-scope form.

(In Examples 2 and 3, the letters P and Q should also be universally quantified in front of the formula. We have omitted that here to avoid irrelevant clutter.)

Example 3. $\forall a\ (\ \forall s\ \exists t\ (Q(s,t) \rightarrow P(s,t)) \rightarrow \exists x\ \ P(a,x))$

The skolemized form is

$$(\overbrace{[Q(s,t_s) \rightarrow P(s,t_s)]}^{\alpha} \Longrightarrow P\ (a_0,x))$$

We are using shorthand t_s for the skolem expression (ts s), and a_0 for (aS).

If we attempt to prove $P(a_0,x)$ by backchaining on α with a_0/s, t_{a_0}/x, we get the subgoal

$$A:\ \ (\ [Q(s,t_s) \rightarrow P(s,t_s)] \Longrightarrow Q(a_0,t_{a_0}))$$

and it is A that we wish to unskolemize.

L = (s (tS s) (aS) (tS (aS)))

LL = ((tS s))

L' = (s (aS))

(SORT-SKO L') = ((aS) s)

(RECON-SKO A T L')

$= \forall a\ (\ \forall s\ \ [Q(s,\ (tS\ s)) \rightarrow P(s,\ (tS\ s))] \rightarrow Q(a,\ (tS\ a))),$

and

(RECONSTRUCT-QUANTIFIERS A)

$= \forall tS\ \forall a(\qquad\qquad\qquad\qquad).$

Notice that now we have universally quantified the <u>function</u> variable tS, out in front of the whole formula. This was required because tS takes on different arguments, s and a, at different places in the formula.

Example 4. $\forall a \forall b\ (H(a,b) \wedge \forall s\ \exists t\ [Q(s,t) \rightarrow P(s,t)] \rightarrow \exists x\ p(a,x))$.

skolemized:

$$(H(a_0,b_0) \wedge \overbrace{[Q(s,t_s) \rightarrow P(s,t_s)]}^{\alpha} \Longrightarrow P(a,x))$$

Again we want to unskolemize the subgoal A acquired by backchaining on α : a_0/s, t_{a_0}/x.

A: $(H(a_0,b_0) \wedge [Q(s,t_s) \rightarrow P(s,t_s)] \Longrightarrow Q(a_0,t_{a_0})$

(RECONSTRUCT-QUANTIFIERS A)

$= \forall tS\ \forall a\ \forall b\ (H(a,b) \wedge [Q(s, (tS\ s)) \rightarrow P(s,(tS\ s))] \rightarrow P(a, (tS\ a)))$.

Example 5. $\forall a\ \exists y\ (\forall s\ \exists t\ [Q(s,t,y) \rightarrow P(s,t,y)] \rightarrow \exists x\ p(a,x,y))$.

skolemized

$$([Q(s,t_{sy},y) \rightarrow P(s,t_{sy},y)] \Longrightarrow P(a_0x,y)).$$

Again we want to unskolemize the subgoal A acquired by backchaining on α: a_0/s, t_{a_0}/x.

A: $([Q(s,t_{sy},y) \rightarrow P(s,t_{sy},y)] \Longrightarrow P(a_0,t_{a_0y},y))$

(RECONSTRUCT-QUANTIFIERS A)

$= \forall tS\ \forall a\ \exists y([Q(s, (tS\ s\ y),y) \rightarrow P(s,(tS\ s\ y),y)] \rightarrow P(a, (tS\ a\ y),y))$.

The next example, Example 6, is acquired from Example 7 of [3], p. 45, subgoal $(P\to 1\ L_1\ 2\ 0\ P2\to)$. It is in skolemized form.

Example 6. $(a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b) \wedge LUB \wedge L1 \wedge L2$

$$\wedge (s \leq t_x \vee x < s \vee 0 < f(s)) \implies x \leq t_x)$$

Where LUB, L1, L2 are given on pp. 33 of [3]. In particular the t_x of α came from the use of lemma L1, which is

$$\forall x(a \leq x \leq b \wedge 0 < f(x) \to \exists t(t < x \wedge \forall s(t < s \leq x \to 0 < f(s))))$$

which skolemizes as

$$(a \leq x1 \leq b \wedge 0 < f(x1) \to (t_{x1} < x1 \wedge (t_{x1} < s \wedge s \leq x_1 \to 0 < f(s)))).$$

Since in Example 6, t_x appears in α and in the conclusion, and t_{x1} appears in L1, it will be necessary to universally quantify tS outside the whole formula <u>before</u> the quantification of x and x1, instead of quantifying t after x and after x1.

(RECONSTRUCT-QUANTIFIERS EX6) =

$$\forall ts\ \forall f\ \forall a\ \forall b\ \forall \ell\ \exists x$$

$$[a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b) \wedge LUBq \wedge L2q$$

$$\wedge \underbrace{\forall x1(a \leq x1 \leq b \wedge 0 < f(x1) \to (tS\ x1) < x1 \wedge \forall s((tS\ x1) < s \leq x1 \to 0 < f(s)))}_{L1q}$$

$$\wedge \underbrace{\forall s(s \leq (tS\ x) \vee (x < s) \vee 0 < f(s))}_{\alpha}$$

$$\longrightarrow \quad x \leq (tS\ x)],$$

where LUBq and L2q are the fully quantified form of LUB and L2. (See page 33 of [3].)

The next example, Example 7, is a continuation of the last. Instead of backchaining on LUB2 as was (properly) done on page 45 of [3], if we try to backchain on LUB1: b/x, t_b/xL, we obtain the subgoal

$(P \to 1 \quad L_1 \quad 2 \quad \theta \quad P2 \to \quad LUB1)$

Example 7. $(a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b) \wedge LUB \wedge L1 \wedge L2$

$$\wedge\ (s \leq t_x \vee x < s \vee 0 < f(s))$$

$$\longrightarrow f(t_b) \leq 0 \wedge \ell < t_b)$$

This is similar to the last example except that x no longer occurs in the conclusion so the "$\exists x$" in front of the whole formula is changed to a "$\forall x$" in front of α only, and we get

(RECONSTRUCT-QUANTIFIERS EX7) =

$$\forall tS\ \forall f\ \forall a\ \forall b\ \forall \ell$$

$$[a \leq b \wedge f(a) \leq 0 \wedge 0 \leq f(b) \wedge LUBq \wedge L2q$$

$$\underbrace{\wedge \forall x1(a \leq x1 \leq b \wedge 0 < f(x1) \to (tS\ x1) < x1 \wedge \forall s(tS\ \ x1) < s \leq x1 \to 0 < f(s)))}_{L1q}$$

$$\wedge \forall x \forall s(s \leq (tS\ x) \vee (x < s) \vee 0 < f(s))$$

$$\longrightarrow f((tS\ b)) \leq 0 \wedge \ell < (tS\ b)].$$

4. Counterexamples

In proving theorems it is often valuable to show that a certain proposal subgoal is invalid, as, for example, when one tries to use an inappropriate lemma, and a counterexample can often be used to show this invalidity. Such a process was used by Gelernter in his geometry prover [4] and has been proposed and used in a limited way by others. (See [5]).

In actuality what we do when we "give a counterexample", is to prove the negation of the fully quantified theorem. (Thus the motivation for the unskolemization process given in Section 2.) For example, if we want to give a counterexample to the formula

(1) $(\text{Continuous}\ \ f\ [a,b] \rightarrow \exists x(a \leq x \leq b \wedge f(x) = 0))$

we first quantify (1)

(2) $\forall f\ \forall a\ \forall b\ \ (\text{Continuous}\ \ f\ [a,b] \rightarrow \exists x(a \leq x \leq b \wedge f(x) = 0)$

and prove its negation,

(3) $\exists f\ \exists a\ \exists b\ \ (\text{Continuous}\ \ f\ [a,b] \wedge \forall x \sim (a \leq x \leq b \wedge f(x) = 0))$

So finding a counterexample of (1) will, in this case, require finding an instantiation for the variables f, a, and b, in (3). Here, any function f which is continuous but never 0 will suffice, if we give a and b values so that $a \leq b$. (e.g., $a = 0$, $b = 1$, $f = \lambda x 1$).

Notice that (3) is a "higher order" theorem, whereas (1) is first order (because f can be treated as a constant function in (1)). It is often the case that the "counterexample theorem" requires the instantiation of a higher order variable (such as f in the above example), but it also seems to be the case that such a higher order variable is often easy to obtain.

It is this last thought that gives us optimism for automatically generating counterexamples, though we do not address the generation of counterexamples in this paper.

Here we want only to point out the need for unskolemizing a formula before we can look for a counterexample to it. Especially is this essential in cases where the formula, for which we want a counterexample, is a fragment of a suspected theorem or (as was the case in Ex.'s 3-7 of Section 3) a subgoal gotten from a suspected theorem, by using a lemma. Because in these cases (see Ex.'s 3-7) a skolem function can appear with different arguments in different parts of the subgoal, and then universal quantification of that skolem function is required (e.g., tS in Ex.'s 6,7).

Example 7, page 13 is false. To find a counterexample, we first negate its fully quantified form, getting

$$\exists tS\ \exists f\ \exists a\ \exists b\ \exists \ell$$

$$(5)\quad [a \le b \wedge f(a) \le 0 \wedge 0 \le f(b) \wedge \text{LUBq} \wedge \text{L2q}$$
$$\forall x1(a \le x1 \le b \wedge 0 < f(x1) \rightarrow (tS\,x1) < x1 \wedge \forall s((tS\ x1) < s < x1 \rightarrow 0 < f(s)))$$
$$\wedge [0 < f((tS\,b)) \vee (tS\,b) \le \ell]],$$

and prove (5) by giving values to the variables a,b,ℓ, and the variable functions tS and f. For example, the following values will satisfy (5):

$$a = 0,\quad b = 1,\quad \ell = 0,$$
$$f = \lambda x\, x,\quad tS = \lambda x\, \frac{x}{2}\ .$$

Because, then (3) becomes

$$[0 \le 1 \wedge 0 \le 0 \wedge 0 \le 1 \wedge \text{TRUE}^{*} \wedge \text{TRUE}^{*}$$
$$\wedge \forall x1(0 \le x1 \le 1 \wedge 0 < x1 \rightarrow \frac{x1}{2} < x1 \wedge \forall s(\frac{x1}{2} < s \le x1 \rightarrow 0 < s))$$
$$\wedge [0 < \frac{1}{2} \vee \frac{b}{2} \le 0]],$$

which is true.

*These are automatically true for continuous functions.

A "higher order prover" such as those of Huet [6], Andrews [7], and Darlington [8], can, in theory, prove such theorems but as yet their power is limited. We believe that special automatic procedures can be developed to efficiently handle a sizable fraction of these higher-order theorems that arise as "counterexample theorems". Our efforts on this will be the subject of another paper.

References

1. W. W. Bledsoe and Mabry Tyson. The UT Interactive Theorem Prover. The Univ. of Texas Math. Dept. Memo ATP-17A, June 1978.

2. W. W. Bledsoe. A Maximal Method for Set Variables in Automatic Theorem Proving. The Univ. of Texas Math. Dept. Memo ATP-33A, July 1977. Proc. IJCAI-77, MIT, Aug. 1977, pp. 501-510. To appear in MI-9.

3. W. W. Bledsoe, Peter Bruell, and Robert Shostak. A Prover for General Inequalities. The Univ. of Texas Math. Dept. Memo ATP-40, June 1978.

4. H. Gelernter. Realization of a Geometry Theorem-proving Machine. Proc. Int. Conf. Information Processing, Paris UNESCO House (1959) 273-282.

5. A. Reiter, A Semantically Guided Deductive System for Automatic Theorem Proving. Proc. Third Int. Joint Conf. Artificial Intelligence (1973) 41-46; IEEE Trans. on Elec. Computing C-25 (1976) 328-334.

6. G. P. Huet. Experiments with an Interactive Prover for Logic with Equality, Report 1106. Jennings Computing Center, Case Western Reserve University

7. Peter Andrews. Theorem Proving in Type Theory. Proc. IJCAI-77, p.566.

8. Jared Darlington. Deductive Plan Formation in Higher Order Logic. Machine Intelligence 7, pp. 129-137.