# University of Waterloo

*INVOICE*

Waterloo, Ontario, Canada
N2L 3G1

Faculty of Mathematics
Department of Computer Science
519/885-1211

Telex Number
069-55259

September 10, 1986.

EXON Production Research Company,
Information Centre,
P.O. Box 2189,
Houston, Texas. 77252

Attn: D. McDonald - Re: Order No. 65358

received payment OCT 15 1986 B.D.

Dear Sir or Madam:

Enclosed please find a copy of our technical report CS-80-01 which was requested by phone on September 9th.

Please be advised that the price for reprinting this report is $3.74 Canadian. Would you please make your cheque or money order payable to the University of Waterloo, Computer Science Department.

Thank you for your interest in our department.

Yours truly,

Susan DeAngelis

Susan DeAngelis (Mrs.),
Technical Report Secretary.

/sd
Encl.

796-0015A

# REQUEST FOR PAYMENT

**EXXON PRODUCTION RESEARCH COMPANY**
P. O. Box 2189
Houston, Texas 77252-2189

DATE: 10-1-1986

VOUCHER NUMBER:

[x] **VOUCHER REQUEST** - Two signatures required (unless initiated by an individual authorized to approve without additional signatures). Should be initiated by an employee authorized to sign and approved by a direct supervisor authorized to approve as indicated on the authorized signature list.

[ ] **EMPLOYEE EXPENSE ADVANCE** - To be signed by the employee requesting advance expense funds and approved by a direct supervisor authorized to perform this function.

DIVISION: Controllers

SECTION: Payables

ACCOUNTING OFFICE OR AFFILIATE TO BE CHARGED: EPR

PURPOSE:

To pay foreign currency invoice # 9/10/86 in Canadian dollars only!

3.74 Can $

Rate - .7246 US - $ 2.71

FOREIGN CURRENCY

| AMOUNT 9-18 | MAIN ACCT. 29-31 | DIV. 32-33 | SUB. ACCT. 34-37 | DETAIL 38-44 | SUB. CODE 45-48 | SOC. SEC. NO. OR CONTROL NUMBER 49-57 | LAST NAME OR DESCRIPTIVE DATA 58-72 | MO. 73-74 | TAX 75 |
|---|---|---|---|---|---|---|---|---|---|
| | 38 | 34 | 10 | 2702 | 1312 | | Univ. of Waterloo | 10 | 1 |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

*AUTHORIZED APPROVAL SHOULD BE ON PAGE 2.*

TO BE PAID BY: [ ] CURRENCY [ ] CHECK- CHECK NUMBER

AMOUNT PAYABLE (WRITTEN OUT): Two & 71/100 DOLLARS

AMOUNT PAYABLE: $ 2.71

NAME AND ADDRESS OF PAYEE

University of Waterloo
Department of Computer Science
Waterloo Ontario
Canada N2L 3G1

NAME AND ADDRESS FOR DELIVERY TO OTHER THAN PAYEE
(SPECIAL APPROVAL AUTHORITY REQUIRED)

# Printing Requisition/Graphic Services

18999

1. **Please complete unshaded areas** on form as applicable.
2. **Distribute copies** as follows: White and Yellow to Graphic Services. **Retain Pink Copies for your records.**
3. On completion of order the **Yellow copy will be returned** with the printed material.
4. **Please direct enquiries, quoting requisition number and account number,** to extension 3451.

TITLE OR DESCRIPTION: CS-80-01

DATE REQUISITIONED: May 17
DATE REQUIRED: May 18
ACCOUNT NO.: 126443102

REQUISITIONER – **PRINT**
PHONE: 2192
SIGNING AUTHORITY: Sue DeAngelis

**MAILING INFO –** NAME / DEPT. / BLDG. & ROOM NO.
☐ DELIVER
☑ PICK-UP



NUMBER OF PAGES: 29
NUMBER OF COPIES: 1

TYPE OF PAPER STOCK: ☑ BOND ☐ NCR ___ PT. ☐ COVER ☐ BRISTOL ☐ SUPPLIED ☐

PAPER SIZE: ☑ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17 ☐

PAPER COLOUR: ☑ WHITE ☐
INK: ☑ BLACK ☐

PRINTING: ☐ 1 SIDE ___ PGS. ☑ 2 SIDES ___ PGS.
NUMBERING: FROM ___ TO ___

BINDING/FINISHING: ☑ COLLATING ☑ STAPLING ☐ ___ HOLE PUNCHED ☐ PLASTIC RING

FOLDING/PADDING
CUTTING SIZE

**Special Instructions**

S

**COPY CENTRE** — OPER. NO. / BLDG. / MACH. NO.

**DESIGN & PASTE-UP**

| OPER. NO. | TIME | LABOUR CODE |
|---|---|---|
| | | D01 |
| | | D01 |
| | | D01 |

**TYPESETTING**

| | QUANTITY | | | |
|---|---|---|---|---|
| PAP00000 | | | | T01 |
| PAP00000 | | | | T01 |
| PAP00000 | | | | T01 |

**PROOF**

| | | | | |
|---|---|---|---|---|
| PRF | | | | |
| PRF | | | | |
| PRF | | | | |

**NEGATIVES**

| | QUANTITY | OPER. NO. | TIME | LABOUR CODE |
|---|---|---|---|---|
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |

**PMT**

| | | | | |
|---|---|---|---|---|
| PMT | | | | C01 |
| PMT | | | | C01 |
| PMT | | | | C01 |

**PLATES**

| | | | | |
|---|---|---|---|---|
| PLT | | | | P01 |
| PLT | | | | P01 |
| PLT | | | | P01 |

**STOCK**

| | | | | |
|---|---|---|---|---|
| | | | | 001 |
| | | | | 001 |
| | | | | 001 |
| | | | | 001 |

**BINDERY**

| | | | | |
|---|---|---|---|---|
| RNG | | | | B01 |
| RNG | | | | B01 |
| RNG | | | | B01 |
| MIS00000 | | | | B01 |

**OUTSIDE SERVICES**

$ ___ COST

TAXES – PROVINCIAL ☐ FEDERAL ☐

# ON REVERSE SKOLEMIZATION

by

P.T. Cox
Department of Computer Science
University of Auckland, New Zealand

T. Pietrzykowski
Department of Computer Science
University of Waterloo

Research Report CS-80-01
January 1980

ON REVERSE SKOLEMIZATION

by

P.T. Cox

T. Pietrzykowski

ABSTRACT

An algorithm is presented which, for an arbitrary literal containing Skolem functions, outputs a set of closed quantified literals with the following properties. If a and b are formulae we define $a \supset b$ iff $\{sk(a), dsk(b)\}$ is unifiable where sk denotes Skolemization and dsk denotes the dual operation, with the roles of $\forall$ and $\exists$ reversed. If d is an arbitrary literal and X is the output, then:

(i) Soundness: if $x \in X$ then $x \supset d$

(ii) Completeness: if $a \supset d$ then exists $x \in X$ such that $a \supset x$

(iii) Nonredundancy: if $x, y \in X$ then $x \not\supset y$ and $y \not\supset x$ .

## 1: Introduction

We consider the problem of reversing Skolemization and present an algorithm which assigns to a literal one or more closed literals where here, as in the rest of this paper, "closed literal" means a closed formula whose matrix is a literal. In the simplest case, if the input literal is the result of skolemizing a closed literal then by applying our algorithm, skolemizing and applying the algorithm again we will produce the original closed literal.

In the general case, however, the situation is more complex; for example if the input literal is one deduced by a mechanical question answering system. The ability to quantify such literals is especially important when the system attempts to answer a question beginning "Why ..." [ 2]. For such applications, the output of our algorithm must have properties of completeness and implicational independence. By "completeness" we mean that if some closed literal A implies the input literal B in a general sense to be defined later, then there is an output C from our algorithm such that A implies C . By "implicational independence" we mean that no output implies another different output.

## 2: Preliminaries

In this section we review standard concepts and notation, as well as introduce some specific definitions.

2.1: We shall use the word expression to refer to literals, terms and variables, where a variable is not a term.

Any term beginning with a Skolem function is called a Skolem term.

A quantifier string is a string of the from $Q_1x_1 \ldots Q_nx_n$ $(n \geq 0)$ where $Q_i$ is either $\exists$ or $\forall$ $(1 \leq i \leq n)$ and $x_1,\ldots,x_n$ are distinct variables.

We use the word "formula" with its standard meaning in mathematical logic.

If $s = pm$ is a formula such that $p$ is a quantifier string and $m$ contains no quantifiers then we define:

$$\text{prefix}(s) = p$$

$$\text{matrix}(s) = m$$

If $a$ is any string, the head of $a$ is the leftmost symbol of $a$.

If $a = x(t_1,\ldots,t_n)$, $b = x(s_1,\ldots,s_n)$ are expressions, then expressions $t$ and $s$ are said to be vis-a-vis in $a$ and $b$ iff for some $i$ $(1 \leq i \leq n)$ wither $s=s_i$ and $t=t_i$, or $t$ and $s$ are vis-a-vis in $t_i$ and $s_i$.

If $m$ is a literal, $p$ is a quantifier string and $v$ is a variable which does not occur in $p$, we define:

$$sk(m) = m$$

$$sk(p\forall vm) = sk(pm)$$

$$sk(p\exists vm) = sk(pm\theta)$$

where $\theta = \{v \leftarrow f(u_1,\ldots,u_r)$, $f$ is a new Skolem function and $u_1,\ldots,u_r$ are all the variables immediately preceded by $\forall$ in $p$. We also define a function dsk with the same range as sk by replacing in the above definition "sk" by "dsk", "$\forall$" by "$\exists$", and "$\exists$" by "$\forall$". Clearly, sk is Skolemization and dsk is the dual operation (see [3]).

If $m$ is a formula and $b$ is an occurrence of an expression in $m$, then $b$ is called a top-level occurrence (in m) iff it is not a proper subexpression of a Skolem subterm (of m).

If m is a formula and b is an expression with a top-level occurrence in m , then we will say that b is top-level (in m).

We will abbreviate the phrase "top-level Skolem" to TS.

If x is an expression we wrote x[t] to indicate that an expression t occurs in x .

## 2.2: Unification

We assume that the reader is familiar with standard definitions of such concepts as "substitution", "variant", "unification." Substitutions will be denoted by lower case Greek letters. We will call a substitution that transforms a literal into one of its variants a renaming. We abbreviate the phrase "most general unifier" to mgu. We assume a familiarity with some of the standard terminology of graph theory, used only in the proof of theorem 4.3.2.

We extend the definition of unification as follows: $\mathcal{E}$ is a set of sets of expressions and $\theta$ is a substitution, $\theta$ unifies $\mathcal{E}$ if and only if $\theta$ unifies E for each $E \in \mathcal{E}$.

The following results is used in section 4.

### 2.2.1: Lemma

If X and Y are sets of expressions or sets of sets of expressions, then $X \cup Y$ is unifiable iff X is unifiable and $Y\sigma$ is unifiable where $\sigma$ is an mgu of X .

### 2.2.2: Baxter's unification algorithm

Here we present a unification algorithm due to Baxter, which is used only in the proof of theorem 4.3.2. The result otherwise is only available as a technical report [1].

Let C be a set of unordered pairs of expressions. If F is a partition of the set of subexpressions of C, and p and q are subexpressions of C, then we denote by $[p]_F$ the class in F which contains p. When F is understood from the context, we will write [p] for $[p]_F$.

In the following, $F_n$ is the partition of all subexpressions of C in which each class contains a single expression.

```
algorithm TRANSFORM(C)

S ← C
P ← F.
while S ≠ ϕ

  do { Delete a constraint {p1,p2} from S
       if [p1] ≠ [p2]
       then { if [p1] contains a term f1(q11,...,q1m)
                 and [p2] contains a term f2(q21,...,q2n)
              then { if f1 ≠ f2
                     then { unification fails
                            stop
                     else add to S the pairs:
                          {q11,q21},...,{q1n,q2n}
              Replace [p1] and [p2] by [p1] ∪ [p2] in F

TRANSFORM ← F

stop
```

This algorithm detects nonunifiability due to conflict of terms. It remains to detect nonunifiability of the type characterized by the pair $\{x,f(x)\}$.

The unification graph for C is a directed graph whose vertex set is TRANSFORM(C) (if this is defined; that is TRANSFORM has not detect nonunifiability). For each pair of vertices X and Y , (X,Y) is an edge iff p is a subexpression of g , where $q \in X$ and $p \in Y$ .

2.2.2.1: Lemma: A set of apirs of expressions C is unifiable iff TRANSFORM(C) succeeds and the unification graph for C has no cycles.

## 3: The algorithms

In this section we describe two algorithms which together produce the required set of closed literals. The first of these algorithms, PREPROCESS, is unnecessary in the case when the input literal has no unifiable TS subterms.

If $\sigma$ is a substitution, m is a literal, t is a top-level subexpression of m and x is a top-level variable of m , we say that $\sigma$ disturbs t in m with x iff x occurs in t but not in $t\sigma$ , or x occurs in $t\sigma$ but x not in t . We will omit "in m" and "with x" when m is understood from context, and x is irrelevant. We say that $\sigma$ disturbs m iff $\sigma$ disturbs some top-level subexpression of m .

Let d be a literal.

```
algorithm PREPROCESS(d)
W ← {d}
R ← ϕ
while W ≠ ϕ
   do  delete e from W;
       if  { every unifier of every pair of distinct TS subterms of
           { e disturbs e , and R contains no variant of e
       then R ← R ∪ {e}
       W ← W ∪ { eσ|t and s are distinct TS subterms of e with mgu σ}
PREPROCESS ← R
stop
```

If $m$ is a formula, we define:

$\mathcal{W}(m) = \{\{v \mid v$ is a free top-level variable of $m$ and occurs in $t\}$
$\mid t$ is a TS subterm of $m\}$

We then define free(m) as the set of lower bounds of the set $\mathcal{W}(m)$ with the partial ordering $\subseteq$ . For example, if $m = \exists w P(\alpha(x,w),\beta(y,z,u),\gamma(x,\delta(y),z),x,y,z,w)$, where all the function symbols are Skolem, then $\mathcal{W}(m) = \{\{x\},\{y,z\},\{x,y,z\}\}$ and free(m) = $\{\{x\},\{y,z\}\}$.

If $m$ is a formula, we define:

ground(m) = $\{t \mid t$ is a TS subterm of $m$, and contains no free top-level variables of $m\}$

If $X$ is a set, we denote by $\vec{X}$ an arbitrary but fixed ordering of $X$ .

If $\vec{X} = (x_1,\ldots,x_n)$ we define:

$$\forall(\vec{X}) = \forall x_1 \forall x_2 \ldots \forall x_n \qquad (n \geq 0)$$
$$\exists(\vec{X}) = \exists x_1 \exists x_2 \quad \exists x_n \qquad (n \geq 0)$$

If $X$ is a set of variables and $D$ is a set of variables or terms such that $|X| = |D|$, and if $\vec{X} = (x_1,\ldots,x_n)$ and $\vec{D} = (g_1,\ldots,g_n)$, then we denote the substitution $\{x_1 \leftarrow g_1,\ldots,x_n \leftarrow g_n\}$ by $\{\vec{X} \leftarrow \vec{D}\}$.

If $m$ is a formula or expression, and $a$ and $b$ are expressions, then repl(a,b,m) is the formula or expression obtained by replacing all top-level occurrences of $a$ in $m$ by $b$ . We extend this definition to ordered sets of expressions as follows:

$$\text{repl}((a_1,\ldots,a_n),(b_1, \quad ,b_n),m)$$
$$= \text{repl}((a_1,\ldots,a_{n-1}),(b_1,\ldots,b_{n-1}),\ \text{repl}\ (a_n,b_n,m))$$

Let us note that if $u$ is a free, top-level variable of $m$ then $\text{repl}(u,b,m) = m\{u \leftarrow b\}$.

Now we shall present a second main algorithm:

Let d be a literal.

```
algorithm QUANTIFY(d)
Q ← φ
S ← {d}
while S ≠ φ
   do  { delete s from S ;
       { G ← ground(s)
       { if   { for every mgu σ of every pair of distinct
       {      {   terms in G
       {      { either σ disturbs a top-level variable in s
       { (#)  { or  for some v and y , where v is the
       {      {     new variable corresponding to some TS
       {      {     subterm t of d , ∀v occurs in prefix(s)
       {      {     to the left of ∃y and σ disturbs t with
       {      {     y .
       { then { F ← free(s)
       {      { H ← {v|v is a free variable in s , and
       {      {       does not occur in any TS subterm of s}
       {      { p ← prefix(s)∀(V)     (see * below)
       {      { m ← repl(G,V,matrix(s))
       {      { if  F = φ
       {      { then  Q ← Q ∪ {p∃(H)m}
       {      { else   { while F ≠ φ
       {               {    do  {delete F from F :
       {               {        {S ← S ∪ {p∃(F)m}
QUANTIFY ← Q
stop
```

(* V is a set of variables which do not occur in s , and |V| = |G|.)

Examples which illustrate these algorithms are presented in section 5 where they may be more fully appreciated in the light of the results presented in section 4.

## 4: Correctness and Implicational Independence

Here we prove some properties of the algorithms considered independently, such as their termination; and some properties of the algorithms combined.

### 4.1: Termination

4.1.1: Lemma: PREPROCESS(d) halts for any literal d.

Proof: If a is a literal, we define:

$$f(a) = (n!)^2$$

where n = number of distinct TS subterms of a.

If W is a set of literals, we define:

$$F(W) = \sum_{a \in W} f(a)$$

Let W' and W" be the value of W at the beginning and end of some execution of the loop; let a be the element of W' deleted and $a_1, \ldots, a_k$ the literals added, then:

$$W'' = W' - \{a\} \cup \{a_1, \ldots, a_k\}$$

$$\therefore\ F(W'') = F(W') - f(a) + \sum_{i=1}^{k} f(a_i)$$

If $f(a) = (n!)^2$, then $k \le \frac{1}{2}n(n-1)$ and $f(a_i) \le ((n-1)!)^2$.

$$\therefore\ F(W'') \le F(W') - (n!)^2 + \tfrac{1}{2}n(n-1)((n-1)!)^2 < F(W').$$

Since F(W) is always non-negative, CASES obviously halts. □

4.1.2: Lemma: QUANTIFY(d) halts for any literal d.

Proof: We define a non-negative integer-valued function G on sets of formulae as follows:

$$G(s) = \begin{cases} 0 & \text{if } S = \emptyset \\ \sum_{b \in S} N(b)! & \text{otherwise} \end{cases}$$

where N(b) = total number of free variables and Skolem terms in b.

Now consider some execution of the major loop of the algorithm and let S' and S" be the values of S at the beginning and end respectively of this execution. Also let s be the element of S' deleted and $s_1,\ldots,s_k$ the formulae added to S' in the loop. Then:

$$S'' = (S' - \{s\} \cup \{s_1,\ldots,s_k\}$$

and

$$G(S'') = G(S') - N(s)! + \sum_{j=1}^{k} N(s_j)!$$

If s has no free variables occurring in Skolem terms, then $F = \emptyset$ so that $k=0$ and:

$$G(S'') = G(S') - N(s)!$$
$$< G(S')$$

If s has free variables occurring in Skolem terms, then $N(s) \geq N(s_j) + 2$ $(1 \leq j \leq k)$ since at least one Skolem term and one free variable of s is quantified in $s_j$. Also $k \leq N(s)$, so that:

$$f(S'') \leq f(S') - N(s)! + k(N(s)-2)!$$
$$\leq f(S') - N(s)! + N(s)(N(s)-2)!$$
$$< f(S')$$

Since G(S') is non-negative, the algorithm must halt. □

4.2: Soundness

If a and b are formulae whose matrices are literals we write $a \supset b$ iff {sk(a), dsk(b)} are unifiable. In the case when a and b are closed, our definition coincides with the standard definition of $\supset$.

4.2.1: Lemma: Soundness of QUANTIFY.

If $a \in$ QUANTIFY(d), then $a \supset d$.

Proof: We will show that at the beginning of every execution of the major loop, if $b \in S \cup Q$ then {sk(b), d} has a unifier $\sigma$ such that no free variable of b occurs in $\sigma$.

At the beginning of the first execution of the major loop $S \cup Q = \{d\}$, so the result clearly holds.

Let S', Q' and S", Q" be the values of S and Q at the beginning and end respectively, of some execution of the major loop. Assume the result holds for $S' \cup Q'$. Now suppose $b \in S'' \cup Q''$ then either $b \in S' \cup Q'$ in which case the result holds, by the above assumption; or b is introduced during the current execution of the loop. Let s be the element of S' deleted then:

$$b = p'\forall(\vec{V})\exists(\vec{F})\ \mathrm{repl}(\vec{G},\vec{V},m')$$

where p' = prefix(s), m' = matrix(s), and V, F and G are as defined in the algorithm.

$$\text{Let } c = p'\forall(\vec{V})\mathrm{repl}(\vec{G},\vec{V},m')$$

$$\begin{aligned}\text{Then } sk(c) &= sk(p'\mathrm{repl}(\vec{G},\vec{V},m')) \\ &= \mathrm{repl}(\vec{G},\vec{V},sk(p'm')) \\ &= \mathrm{repl}(\vec{G},\vec{V},sk(s))\end{aligned}$$

Since $s \in S' \cup Q'$, by the above assumption {sk(s), d} has a unifier $\sigma$ containing no free variables of s.

$$\text{Let } \gamma = (d\gamma)\ \{\vec{V}\leftarrow\vec{G}\} \circ\sigma$$

$$\begin{aligned}\text{then } sk(c)\gamma &= \mathrm{repl}(\vec{G},\vec{V},sk(s))\gamma \\ &= sk(s)\sigma \\ &= d\sigma \\ &= d\gamma \quad \text{since none of the variables in } V \text{ occur in } d.\end{aligned}$$

Hence $\gamma$ unifies {sk(c), d}; also, since the elements of G are ground Skolem terms of s, $\gamma$ contains no variables free is s. Now let p" = prefix(c), m" = matrix(c);

then $b = p''\exists(\vec{F})m''$

$\therefore sk(b) = sk(p''repl(\vec{F},\vec{T},m''))$

where T is a set of Skolem terms containing only variables immediately preceded by $\forall$ in p"

$= repl(\vec{F},\vec{T},sk(p''m''))$

$= repl(\vec{F},\vec{T},sk(c))$

Let $\delta = \{\vec{F} \leftarrow \vec{T}\} \circ \gamma$

$= \gamma \circ \{\vec{F} \leftarrow \vec{T}\}$

since none of the variables in F occur in $\gamma$.

(Note that the meaning of $\vec{T}\gamma$, although we have not defined it, is obvious.)

Then $d\delta = (d\gamma)\{\vec{F} \leftarrow \vec{T}\}$

$= (sk(c)\gamma)\{\vec{F} \leftarrow \vec{T}\gamma\}$

$= (sk(c)\{\vec{F} \leftarrow \vec{T}\})\gamma$

$= repl(\vec{F},\vec{T},sk(c))(\{\vec{F} \leftarrow \vec{T}\}\circ\gamma)$

$= sk(b)\delta$

Hence $\delta$ unifies $\{sk(b), d\}$ and does not contain any variables free in b. Note that we have assumed that $F \neq \phi$. In the case when $F = \phi$ and $H \neq \phi$, b=c in the above, and only the proof that $\{sk(c), d\}$ is unifiable is necessary. This can be obtained from the above by replacing all occurrences of F by H. □

<u>4.2.2: Theorem</u>: Soundness

If $b \in$ QUANTIFY(c), where $c \in$ PREPROCESS(d), then $b \supset d$.

The proof follows immediately from lemma 4.2.1, and lemma 4.4.2.

## 4.3: Completness

If $d$ and $d'$ are literals such that $d' \in$ PREPROCESS(d), we shall denote by $P_{d'}$, the partition of the set of all TS subterms of $d$ such that $s, t \in X \in P_{d'}$, iff $s' = t'$, where $s'$ and $t'$ are TS subterms of $d'$ which are vis-a-vis $s$ and $t$ respectively.

4.3.1: Lemma: Completeness of PREPROCESS

If $c$ is a closed literal such that $c \supset d$, then there is a literal $b \in$ PREPROCESS(d) such that $\{sk(c), b\}$ has a unifier $\xi$ with the property that $t\xi \neq s\xi$ for all pairs of distinct TS subterms $t$ and $s$ of $b$.

Proof: Let $\mu$ be an mgu of $\{sk(c), d\}$; denote $sk(c)\mu$ by $e$; let $\eta$ be an mgu for $P_e$ ; and let $d' = d\eta$. By lemma 2.2.1, $sk(c)$ and $d'$ are unifiable; let their mgu be $\sigma$. It is easy to show that at some time during the execution of PREPROCESS(d), $d' \in W \cup R$: to show this, we can select pairs of terms which belong to the same class of $P_e$ ; these terms are obviously unifiable, and will be unified during some execution of the loop. By continuing this process we can construct a sequence of literals such that each is in $W \cup R$, and the last in the sequence is $d'$. If $d' \in R$, then $b = d'$ is obviously the required literal.

If $d' \notin$ PREPROCESS(d), we construct a sequence $d_1, d_2, \ldots, d_n$ $(n \geq 2)$ where $d_1 = d'$, and $d_{i+1} = d_i\theta_i$ where $\theta_i$ $(1 \leq i \leq n-1)$ is an mgu of some pair of distinct TS subterms of $d_i$ which does not disturb $d_i$, and $d_n$ has no pair of distinct TS subterms with a unifier $\theta_1$ that does not disturb $d_1$. Also since the number of distinct unifiable TS subterms is reduced at each extension of the sequence, the construction must terminate. Obviously $d_n \in$ PREPROCESS(d).

It remains only to show that $d_n$ and sk(c) are unifiable. We will show that $d_2$ and sk(c) are unifiable; the argument can be extended to the rest of the sequence.

Denote $\theta_1$ by $\theta$. We will show that $\sigma\circ\theta\circ\sigma\circ\theta\circ\sigma$ is a unifier of $d_2$ and sk(c) with the required property. First we show that $\sigma\circ\theta\circ\sigma\circ\theta\circ\sigma = \theta\circ\sigma\circ\theta\circ\sigma\circ\theta\circ\sigma$

(i) If v is a top-level variable, then $v = v\theta$ since no replaced variables of $\theta$ are top-level because $\theta$ does not disturb $d_1$. In this case the result obviously holds.

(ii) If v is not top-level then $v = v\sigma$, since all the replaced variables of $\sigma$ are top-level because $\sigma$ does not unify any distinct TS subterms of $d_1$, and is an mgu.

(a) If $v\theta$ contains no top-level variables then:

$$v\theta = v\theta\sigma$$

$$v\theta\sigma\theta\sigma\theta\sigma = v\theta\theta\sigma\theta\sigma$$

$$= v\theta\sigma\theta\sigma$$

since none of the replaced variables of $\theta$ occur in the terms of $\theta$.

$$= v\sigma\theta\sigma\theta\sigma$$

(b) If $v\theta$ contains a top-level variable, say u, we will show that v does not occur in $u\sigma$. Suppose the contrary; then in the simplest case there is an expression t in sk(c) which is vis-a-vis u, and in which there occurs a variable x, which is top-level in sk(c) (as are all the variables of sk(c)) and is vis-a-vis a term s in $d_1$, where v occurs in s. Since $\theta$ does not disturb $d_1$, u must also occur in s. Therefore, to unify $d_1$ and sk(c), it is necessary to unify {{u,t[x]} {s[u],x}}, which is impossible. In the general case, this nonunifiability between u and s will always arise, although there may be more than two pairs of expressions contributing to it. Now if y is <u>any</u> variable with the same properties as v, then by identical reasoning, y does not occur in $u\sigma$. Therefore, $v\theta\sigma$ does not contain any variables of this type, and no top-level variables. So by similar reasoning to case (a), the result holds.

Hence $\sigma\circ\theta\circ\sigma\circ\theta\circ\sigma$ is a unifier of $d_1\theta$ and sk(c). The fact that this substitution does not unify any distinct TS subterms of $d_1\theta$ follows from the fact that $\sigma$ does not unify any distinct TS subterms of $d_1$. □

<u>4.3.2: Theorem</u>: Completeness

If a is a closed literal, and d is a literal such that $a \supset d$, there exists c and b such that $c \in$ PREPROCESS(d), $b \in$ QUANTIFY(c), and $a \supset b$.

<u>Proof</u>: By lemma 4.3.1, there is a literal $c \in$ PREPROCESS(d) such that {sk(a), c} has an mgu that does not unify any distinct TS subterms of c. We now consider the execution of QUANTIFY(c), and show that at the beginning of every execution of the major loop there is a formula $e \in S \cup Q$ such that

(i) {dsk(e), sk(a)} is unifiable

(ii) no vertex of the unification graph of {dsk(e), sk(a)} contains more than one Skolem subterm of dsk(e).

(iii) either $e \in Q$

or if v is a variable existentially quantified in e and t is a TS subterm of e, then there is no walk from [v] to [t] in the unification graph $U_e$ of {dsk(e), sk(a)}.

At the beginning of the first execution of the major loop, $e = d \in S \cup Q$ clearly satisfies the conditions.

Let $S'$, $Q'$ and $S''$, $Q''$ be the values of $S$ and $Q$ at the beginning and end respectively of some execution of the major loop. Assume the result holds for $S' \cup Q'$ and let $e \in S' \cup Q'$ be the formula with the required properties; then either $e \in S' \cup Q''$ in which case the result holds for $S'' \cup Q''$ or $e$ is the formula deleted from $S'$ in the execution of the loop. In the latter case we assume that the condition of the first <u>if</u> statement is satisfied so that quantification of $e$ proceeds: we will show that a formula $f$ with the required properties is added to $S' \cup Q'$ during the execution of the loop.

Let $f = p\forall(\vec{V})\exists(\vec{K})\ \mathrm{repl}(\vec{G},\vec{V},m)$ where $p$, $m$, $G$, $V$ are as defined in the algorithm and $K$ is either $F$ or $H$ as defined in the algorithm: then $f \in S'' \cup Q''$. The particular $K$ we choose for constructing $f$ is irrelevant to the proof that $f$ satisfies conditions (i) and (ii); consequently we will postpone the explanation of how $K$ is selected until these conditions have been proved.

$$\begin{aligned} \text{Now}\quad \mathrm{dsk}(f) &= \mathrm{dsk}(p\forall(\vec{V})\ \mathrm{repl}(\vec{G},\vec{V},m)) \\ &= \mathrm{dsk}(p\ \mathrm{repl}(\vec{V},\vec{G}',\mathrm{repl}(\vec{G},\vec{V},m))) \\ &= \mathrm{dsk}(p\ \mathrm{repl}(\vec{G},\vec{G}',m)) \\ &= \mathrm{repl}(\vec{G},\vec{G}',\mathrm{dsk}(e)) \end{aligned}$$

where $G'$ is a set of new Skolem terms introduced by the application of dsk.

We now describe the construction of $U_f$ from $U_e$: the reader should verify that this construction is correct. The construction is as follows:

(1) For each $t \in G$ replace $t$ by $t'$ in $[t]$, where $t'$ is the new Skolem term corresponding to $t$.

(2) Delete all vertices which contain expressions which do not occur in dsk(f). Note that such expressions have no top-level occurrences in e, so by condition (ii) on e, these deleted vertices each contain a single expression.

(3) Delete all edges which enter or leave vertices deleted in (2).

(4) For each $t \in G$ and each top-level variable $v$ of dsk(f) which occurs in $t'$, where $t'$ is the new Skolem term corresponding to $t$, add the edge $([t'],[v])$.

We now show that f satisfies the conditions

(i) Suppose there is a closed walk in $U_f$. Since $U_e$ has no closed walks, some of the edges on this walk must be added in the above construction. Suppose there is exactly one such new edge $([t'],[w])$ on the walk, where w is existentially quantified in e and $t'$ is the new Skolem term in dsk(f) corresponding to some term t in e. Then there is a walk from $[w]$ to $[t]$ in $U_e$ contradicting the assumption that e satisfied condition (ii). If the walk contains more than one such edge, we consider the part of the walk connecting two consecutive new edges and obtain the same contradiction. Hence {dsk(f), sk(a)} is unifiable.

(ii) That f satisfies this condition is obvious from the above construction.

(iii) In order to show that f satisfies this condition, we now explain how K is selected. Suppose free(e) $\neq \phi$, then the set $NG = \{t \mid t \text{ is a TS subterm of e and } t \notin \text{ground(e)}\}$

is not empty. Since $U_e$ has no closed walks, it induces a partial ordering $<$ on NG as follows: $t_1 < t_2$ iff there is a walk from $[t_2]$ to $[t_1]$. Let $t$ be a minimal element of NG under this ordering; then we choose $K$ to be that element of free(e) which is a subset of

$\{v|v$ is a free top-level variable of $e$, and occurs in $t\}$.

Now suppose that $v$ is existentially quantified in $f$ and $s$ is a TS subterm of $f$ and hence of $e$. Either $v \in K$, and by the selection of $K$ there is no walk from [v] to [s] in $U_e$; or $v$ is existentially quantified in $e$ so by condition (iii) on $e$, there is no walk from [v] to [s] in $U_e$. Hence if there is such a walk in $U_f$ it must contain at least one new edge introduced in the above construction. Suppose ([s'],[w]) is the last such edge on this walk, then there is a walk in $U_e$ from [w] to [s]; however, $w \in K$ so such a walk contradicts our selection of $K$. Hence no walk from [s] to [v] exists in $U_f$, so condition (iii) is satisfied.

In the case when free(e) $= \phi$, $K = H$ and $f \in Q''$, so condition (iii) holds.

Now suppose that the condition (#) of the first <u>if</u> statement is not satisfied so no further quantification of $e$ is done. Then there is a pair $\{t,s\}$ of distinct TS subterms of $c$ in ground(e) with an mgu $\sigma$ with the following properties:

(i) $\sigma$ does not disturb top-level variables

(ii) for every TS subterm $r$ of $d$ and every variable $y$, if $r$ is disturbed by $\sigma$ with $y$ then either $r$ occurs in $e$ or $\exists y$ occurs in prefix(e) to the left of $\forall x$, where $x$ is the new variable corresponding to $r$.

Let b be the formula added to S when the first such $\exists y$ is introduced into the prefix. Consider that execution of the loop of PREPROCESS(d) in which c is deleted from W: in the loop the literal $c\sigma$ is added to W. Also, unifying any TS subterms of c disturbs c since $c \in$ PREPROCESS(d), therefore, unifying any TS subterms of $c\sigma$ disturbs $c\sigma$; hence $c\sigma \in$ PREPROCESS(d). Now since $\sigma$ disturbs only TS subterms of d that occurs in b it is clear that during the execution of QUANTIFY($c\sigma$) a formula b' will be produced such that:

$$b' = pm'$$

where $p = \text{prefix}(b)$

$$m' = m\sigma$$

$$p = \text{matrix}(b)$$

Suppose that QUANTIFY were modified so that quantification of all formulae is completed (i.e. the first <u>if</u> statement is replaced by its <u>then</u> part). Let g be the formula output by this modified algorithm as a result of complete quantification of b, such that {dsk(g), sk(a)} is unifiable: such a formula exists, by the above proof. The essential difference between b and b' is that some of the TS subterms of b' contain more top-level variables than their counterparts in b. These extra variables, however, will be existentially quantified before any term disturbed with them is removed by QUANTIFY. Consequently, the order of quantifications performed in producing g from b can be exactly duplicated in quantifying b'. The resulting formula g' will differ from g only in the following way: the two quantifiers $\forall x \forall y$ in prefix(g) introduced when quantify removed TS subterms t and s are replaced by a single quantifier $\forall x$ in prefix(g'), and all occurrences of y in matrix(g) are replaced by x in matrix(g'). Obviously {dsk(g'), sk(a)} are unifiable. Note that later in the quantification of b' processing may again be terminated should the condition in the first _if_ statement not be satisfied: in this case we repeat the above construction as many times as necessary. □

4.4: Implicational Independence

4.4.1: Lemma: Implicational Independence of QUANTIFY

If d is a literal, $e,f \in$ QUANTIFY(d), and $e \neq f$, then $e \not\supset f$ and $f \not\supset e$.

Proof: First let us assume that matrix(e) = matrix(f): this assumption is justified by noting that no two formulae in QUANTIFY(d) are variants of each other, and that each TS subterm can always be replaced by the same new variable. Let us also assume that the variables of e and f are ordered by a relation $<$ in an arbitrary but fixed manner, and that blocks of quantifiers of the same type in the prefixes of e and f are arranged according to this ordering: that is, if $Qu_1Qu_2$ occurs in prefix(e) or prefix(f), where Q is $\forall$ or $\exists$, then $u_1 < u_2$.

Since $e \neq f$, prefix(e) $\neq$ prefix(f). Let p be the longest quantifier string which is the left part of both prefixes (note that p could be empty), then:

prefix(e) = pQ'u ...

prefix(f) = pQ"v ...,

where $Q'u \neq Q''v$.

Now if $Q' = Q'' = \forall$, then $u \neq v$. Suppose u and v were introduced to replace TS subterms s and t, respectively; then all top level variables occurring in s and t must occur in p. Therefore, Q"v must occur to the right of $Q'u$ in prefix(e), and is introduced by QUANTIFY at the same time as $Q'u$; hence $u < v$. Similarly, by considering prefix(f), we find that $v < u$. Consequently, not both $Q'$ and $Q''$ are $\forall$. Now suppose that $Q' = \forall$ and $Q'' = \exists$ (or vice-versa); then there is a TS subterm $s$ with all its top-level variables occurring in $p$: QUANTIFY always removes all such terms before existentially quantifying any further top-level variables. This contradicts the supposition that $Q'' = \exists$. The only remaining possibility is that $Q' = Q'' = \exists$, which implies $u \neq v$. Let $U$ be the set of variables existentially quantified at the same time as $u$ in the production of $e$: we define $v$ analgously for $v \Delta f$.

Clearly $U \neq V$, since if $U = V$ we can conclude $u < v$ (from prefix(e)) and $v < u$ (from prefix(f)); also $U \not\subset V$ since $U \subset V$ implies that $V$ cannot be chosen as a minimal set of free variables to be existentially quantified. Consequently, there exists variables $x \in U \backslash V$ and $y \in V \backslash U$, and TS subterms $s$ and $t$ such that $x$ occurs in $s$ not $t$, and $y$ occurs in $t$ not $s$, and:

$$\text{prefix}(e) = p \ldots \exists x \ldots \forall w \ldots \exists y \ldots \forall z \ldots$$

$$\text{prefix}(f) = p \ldots \exists y \ldots \forall z \ldots \exists x \ldots \forall w \ldots$$

where $w$ and $z$ are new variables corresponding to $s$ and $t$ respectively. Hence in order to unify {sk(e), dsk(f)} it is necessary to unify $\{\{\alpha,x\}, \{w,\gamma[y]\}, \{\beta[w],y\}, \{z,\delta[x,y]\}\}$, where $\alpha$ and $\beta$ are Skolem terms introduced by the application of sk to e, and $\gamma$ and $\delta$ are Skolem terms introduced by the application of dsk to f. Therefore, {sk(e), dsk(f)} is not unifiable; by symmetry, neither is {dsk(e), sk(f)}. □

4.4.2: Lemma: If $d' \in$ PREPROCESS(d) and $\Theta$ is an mgu of $P_{d'}$, then $d' = d\Theta$.

The proof follows easily from the definition of PREPROCESS, and is left to the reader.

4.4.3: Corollary: If $d', d'' \in$ PREPROCESS(d) and $P_{d'} = P_{d''}$, then $d' = d''$.

Proof: If $P_{d'} = P_{d''}$ then $d' = d\sigma'$ and $d'' = d\sigma''$ where $\sigma'$ and $\sigma''$ are mgus of $P_{d'}$. Hence either $d''$ and $d'$ are variants, contradicting the definition of PREPROCESS, or $d' = d''$. □

If $d', d'' \in$ PREPROCESS(d), we shall say that $d' > d''$ iff for each $X \in P_{d'}$ there exists $Y \in P_{d''}$ such that $X \subseteq Y$.

4.4.4: Lemma: If $d', d'' \in$ PREPROCESS(d), $d' > d''$, $\sigma'$ is an mgu of $P_{d'}$, and $\Theta$ is an mgu of $P_{d''}\sigma'$ then $d'' = d'\Theta$. Again, the proof, based on the definition of PREPROCESS and lemma 4.4.2, is left to the reader.

4.4.5: Theorem: Implicational Independence

Let d, d', d" be literals such that $d', d'' \in$ PREPROCESS(d) and $d' \neq d''$; and suppose $g' \in$ QUANTIFY(d'), $g'' \in$ QUANTIFY(d"), then:

(i) $g'' \not\Rightarrow g'$

(ii) $g' \not\Rightarrow g''$

Proof: We consider two cases as follows:

(a) Suppose $d' \not> d''$ and $d'' \not> d'$. From the definition of $>$, it follows that d' has TS subterms $s_1', t_1', s_2', t_2'$ which are vis-a-vis $s_1'', t_1'', s_2'', t_2''$ in d", such that $s_1'' = t_1''$ and $s_2' = t_2'$ but $s_1' \neq t_1'$ and $s_2'' \neq t_2''$. Consequently, $\forall x_1''$, $\forall y_1'$ and $\forall x_2'$ occur in prefix(g") where $x_1', y_1''$, $x_2', x_1'', x_2'', y_2''$ are the new variables corresponding to $s_1', t_1', s_2'$

(= $t_2'$), $s_1'$, (= $t_1''$), $s_2''$, $t_2''$ respectively. Therefore, to unify {dsk(g'), sk(g")} it is necessary to unify $\{\{\alpha_1', x_1''\}, \{\beta_1'', x_1''\}\}$ where $\alpha_1'$, $\beta_1'$ are distinct Skolem terms introduced by the application of dsk to g'; this is clearly impossible. Hence (i) is proved, and (ii) is similarly proved by considering the new terms in dsk(g") which are vis-a-vis $x_2$ in sk(g').

(b) Suppose d' > d". Let $\Theta$ be a substitution as defined in lemma 4.4.4 such that d" = d'$\Theta$; by corollary 4.4.3, since d' ≠ d", $P_{d'} \neq P_{d''}$ so $\Theta$ must unify at least one pair of distinct TS subterms of d'. Therefore, vis-a-vis these distinct terms of d' are identical terms of d", so by reasoning identical to that used in case (a), (i) holds.

Since d' $\in$ PREPROCESS(d) but has distinct unifiable TS subterms (since $\Theta$ unifies at least two of them), $\Theta$ must disturb d'. There are two ways this can happen:

<u>either</u> (A) $\Theta$ disturbs a top-level variable u of d'

<u>or</u> (B) $\Theta$ disturbs a top-level subterm of d'.

In case (A), u in d' occurs vis-a-vis a new Skolem term $\alpha$ in sk(g'). Suppose u occurs vis-a-vis an expression t in d". If t is a term, then it occurs vis-a-vis a term in dsk(g") with a head different from that of $\alpha$, so (ii) clearly holds. If t is a variable, say v then it occurs in d" vis-a-vis both u and w in d', where w ≠ u, since $\Theta$ disturbs u. Then v occurs in dsk(g") vis-a-vis two different Skolem terms in sk(g'). Then v occurs in dsk(g") vis-a-vis two different Skolem terms in sk(g'). Again it is clear that (ii) holds.

In case (B) we will show that g' has a TS subterm s' such that:

(1) some top-level variable y' occurs in s" = s'Θ but not in s', and

(2) ∀x' occurs to the left of ∃y' in prefix(g'), where x' is the new variable corresponding to s'.

Let t' and r' be two distinct TS subterms of d' which are unified by Θ. We have two cases to consider:

(a) Suppose t' and r' are not replaced by new variables during the same execution of the major loop of QUANTIFY. Then some top-level variable y' occurs in t" but not in r' and y' is still free. Clearly ∀x' occurs to the left of ∃y' in prefix(g'), where x' is the new variable corresponding to r'.

(b) If t' and r' are replaced by new variables in the same execution of the major loop, then suppose that for all TS subterms s and all top-level variables y and d', if Θ disturbs s with y, ∀x occurs to the right of ∃y in prefix(g') where x is the new variable corresponding to s. In the case when Θ is an mgu of t' and r', processing of the formula by QUANTIFY will be terminated because of condition (#), contradicting the fact that g' ∈ QUANTIFY(d'). The general case, when Θ is not an mgu of r' and t', is left to the reader.

Now for g ∈ QUANTIFY(d"), it follows from (1) that ∃y' occurs to the left of ∀x" in prefix(g'), where x' is the new variable corresponding to s". From this fact, and (2) it follows that to unify {sk(g'), dsk(g")} it is necessary to unify {{x',α[y']},{y',β[x']}} where α and β are Skolem terms introduced by sk and dsk. This proves case (ii). □

## 5: Examples and Final Remarks

First we illustrate QUANTIFY.

5.1: Example: Let d be the literal $P(\alpha(x,y),\beta(x,z),\gamma(y,z),f(x,y),z)$ where $\alpha$, $\beta$ and $\gamma$ are Skolem functions and f is not. Then:

$$\text{QUANTIFY}(d) = \begin{aligned}\{ &\exists x\exists y\forall a\exists z\forall b\forall c\ m,\\ &\exists y\exists z\forall c\exists x\forall z\forall b\ m,\\ &\exists x\exists z\forall b\exists y\forall a\forall c\ m \}\end{aligned}$$

$$\text{where } m = P(a,b,c,f(x,y), z)$$

The reader should note that for every $b \in$ QUANTIFY(d), $dsk(b) \neq d$.

The next example illustrates that expressions which are not top-level have no influence on the output of QUANTIFY; and that the names of Skolem functions are unimportant.

5.2: Example: Let $d_1 = P(f(\alpha),\gamma(g(z)),\beta(x),x)$ and $d_2 = P(f(\beta(z)),\delta(a),\gamma(h(x)),x)$ where $\alpha$, $\beta$, $\gamma$, $\delta$ are Skolem functions, and f is not. Then:

$$\text{QUANTIFY}(d_1) = \text{QUANTIFY}(d_2) = \{\forall y\forall w\exists x\forall v P(f(y),w,v,x)\}.$$

In the preceding examples, no TS subterms are unifiable, so PREPROCESS would have no effect. The next example shows that PREPROCESS is required for completeness.

5.3: Example: Let $d = P(\alpha(z),\alpha(x),x)$, where $\alpha$ is a Skolem function, then:

$$\text{QUANTIFY}(d) = \{\forall y\exists x\forall v P(y,v,x)\} = \{b\}$$

Consider the closed literal $c = \forall y P(y,y,a)$. Clearly $c \supset d$, but $c \not\supset b$. This is because QUANTIFY distinguishes between Skolem terms which are not identical but are unifiable. However:

PREPROCESS(d) = {d,e}

where $e = P(\alpha(x),\alpha(x),x)$.

Then QUANTIFY(e) = $\{\exists x \forall y P(y,y,x)\} = \{f\}$, and $c \supset f$.

Now we provide an example to illustrate how PREPROCESS avoids redundancy.

<u>5.4: Example</u>: Let $d = P(\alpha(x,z),\alpha(x,x),x)$ then PREPROCESS(d) = $\{P(\alpha(x,x),\alpha(x,x),x)\} = \{b\}$. Note that unlike example 5.3, d is not in PREPROCESS(d), since the unification does not cause a disturbance; and that $d \supset b$.

Our final example illustrates the need for the condition (#) in QUANTIFY.

<u>5.5: Example</u>: Let $d = P(\alpha(x),\gamma(y,v),\beta(z,y),\beta(z,x),x,z,v)$ then:

PREPROCESS = {d,e}

where $e = P(\alpha(x),\gamma(x,v),\beta(z,x),\beta(z,x),x,z,v)$.

Suppose condition (#) is removed from QUANTIFY, then this modified algorithm produces from d the following:

$\{\exists x \forall a \exists z \forall b_1 \forall b_2 \exists v \forall c\ m\ (= d_1)$,

$\exists x \forall a \exists v \forall c \exists z \forall b_1 \forall b_2\ m\ (= d_2)$,

$\exists z \forall b_1 \exists v \forall c \exists x \forall a \forall b_2\ m$,

$\exists z \forall b_1 \exists x \forall b_2 \forall a \exists v \forall c\ m$,

$\exists v \forall c \exists x \forall a \exists z \forall b_1 \forall b_2\ m$,

$\exists v \forall c \exists z \forall b_1 \exists x \forall b_2 \forall a\ m\}$

where $m = P(a_1c_1b_1,b_2,x_1z_1v)$

From e, QUANTIFY produces:

$\{\exists x \forall a \exists v \forall c \exists z \forall b\ m'\ (= e_1)$,

$\exists x \forall a \exists z \forall b \exists v \forall c\ m'\ (= e_2)\}$

where $m' = P(a_1c,b_1b_1x_1z_1v)$

Clearly $d_1 \supset e_2$ and $d_2 \supset e_1$. However, condition (#) restricts QUANTIFY such that $d_1$ and $d_2$ are not produced, since at the point where the two unifiable terms have no free variables remaining, the term $\gamma(y,v)$ which is disturbed by the mgu with $x$ is either still in the matrix (in $d_1$) or its corresponding new variable $c$ occurs to the right of $\exists x$ in the prefix (in $d_2$).

## Bibliography

[1] Baxter, L.D., A practical linear unification algorithm. Research Report CS-76-13, Department of Computer Science, University of Waterloo, 1976.

[2] Pietrzykowski, T., Mechanical Hypothesis Formation, Research Report CS-78-33, Department of Computer Science, University of Waterloo, 1978.

[3] Skolem, T., Über die mathematische Logik. Norsk matematisk Tidskrift, 10, pp. 125-142, 1928.

# Printing Requisition/Graphic Services

38573

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: CS-86-01

DATE REQUISITIONED: Jan. 1/86 | DATE REQUIRED: ASAP | ACCOUNT NO.: 1 2 6 9 9 3 1 6 2 4

REQUISITIONER- PRINT: Sue DeAngelis | PHONE: 3172 | SIGNING AUTHORITY: [signature]

MAILING INFO – NAME | DEPT. | BLDG. & ROOM NO. | ☐ DELIVER ☒ PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 21 | NUMBER OF COPIES: 1

TYPE OF PAPER STOCK: ☒ BOND ☐ NCR ___ PT. ☐ COVER ☐ BRISTOL ☐ SUPPLIED ☐

PAPER SIZE: ☒ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17 ☐

PAPER COLOUR: ☒ WHITE ☐ | INK: ☒ BLACK ☐

PRINTING: ☐ 1 SIDE ___ PGS. ☒ 2 SIDES ___ PGS. | NUMBERING FROM TO

BINDING/FINISHING: ☒ COLLATING ☐ STAPLING ☐ HOLE PUNCHED ☐ PLASTIC RING

FOLDING/PADDING | CUTTING SIZE

Special Instructions

COPY CENTRE | OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP | OPER. NO. | TIME | LABOUR CODE

| | | | |
|---|---|---|---|
| | | | D01 |
| | | | D01 |
| | | | D01 |

TYPESETTING | QUANTITY

| | | | | |
|---|---|---|---|---|
| PAP000000 | | | | T01 |
| PAP000000 | | | | T01 |
| PAP000000 | | | | T01 |

PROOF

| | | | | |
|---|---|---|---|---|
| PRF | | | | |
| PRF | | | | |
| PRF | | | | |

NEGATIVES | QUANTITY | OPER. NO. | TIME | LABOUR CODE

| | | | | |
|---|---|---|---|---|
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |
| FLM | | | | C01 |

PMT

| | | | | |
|---|---|---|---|---|
| PMT | | | | C01 |
| PMT | | | | C01 |
| PMT | | | | C01 |

PLATES

| | | | | |
|---|---|---|---|---|
| PLT | | | | P01 |
| PLT | | | | P01 |
| PLT | | | | P01 |

STOCK

| | | | | |
|---|---|---|---|---|
| | | | | 001 |
| | | | | 001 |
| | | | | 001 |
| | | | | 001 |

BINDERY

| | | | | |
|---|---|---|---|---|
| RNG | | | | B01 |
| RNG | | | | B01 |
| RNG | | | | B01 |
| MIS00000 | | | | B01 |

OUTSIDE SERVICES

$ ______ COST

TAXES – PROVINCIAL ☐ FEDERAL ☐

369

# Printing Requisition/Graphic Services

Dept. No. 71969

1. Please complete unshaded areas on form as applicable. (4 part no carbon required).
2. Distribute copies as follows: White, Canary and Pink—Printing, Arts Library or applicable Copy Centre Goldenrod—**Retain.**
3. On completion of order, pink copy will be returned with printed material. Canary copy will be costed and returned to requisitioner, **Retain as a record of your charges.**
4. Please direct enquiries, quoting requisition number, to Printing/Graphic Services, Extension 3451.

**Title or Description:** CS 80-01 Research Report January 1980

**Date:** 15 January /80 **Date Required:** ASAP **Account:** 126- 6297- 41

**Signature:** [signature] **Signing Authority:** T. Pietrzykowski

**Department:** Computer Science **Room:** 5179 **Phone:** 3143 **Delivery:** [X] Mail [ ] Pick-up [ ] Via Stores [ ] Other

**Reproduction Requirements:** [X] Offset [ ] Signs/Repro's [ ] Xerox **Number of Pages:** 29 **Number of Copies:** 30

**Type of Paper Stock:** [X] Bond [ ] Book [ ] Cover [ ] Bristol [ ] Supplied

**Paper Size:** [X] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17

**Paper Colour:** [X] White [ ] Other **Ink:** [ ] Black

**Printing:** [X] 1 Side [ ] 2 Sides **Numbering** to

**Binding/Finishing Operations:** [X] Collating [ ] Corner Stitching [ ] 3 Ring [ ] Tape [ ] Plastic Ring [ ] Perforating

**Folding Finished Size:** 3 STAPLES **Cutting Finished Size**

**Special Instructions:** Could I please have three staples on the left top, middle, and bottom please, with front and backs attached.

| Cost: Time/Materials | Fun. | Prod.Un. | Prod. Cl. | Opr. No. | Mins. | Total |
|---|---|---|---|---|---|---|
| Signs/Repro's | 1 | | | | | |
| Camera | 2 | | | | | |
| Correcting & Masking Negatives | 3 | | | | | |
| Platemaking | 4 | | | | | |
| Printing | 5 | | | | | |
| | | | | | | |
| Bindery | 6 | | 23-10 91-10. | | | |
| | | | | | | |
| Sub. Total Time | | | | | | |
| Sub. Total Materials | | | | | | |
| Prov. Tax | | | | | | |
| Total | | | | | | |

Film Qty Size | Plates Qty Size & Type

Paper Qty Size | Plastic Rings Qty Size

Outside Services

Graphic Services May 68 482-2