STATE UNIVERSITY OF NEW YORK AT BUFFALO

Department of Computer Science

NUMERICAL QUANTIFIERS AND THEIR USE IN REASONING WITH NEGATIVE INFORMATION*

Stuart C. Shapiro

February, 1979

Technical Report Number 153

*This material is based on work supported in part by a Faculty Research Fellowship from the Research Foundation of State University of New York, and in part by the National Science Foundation under Grant No. MCS78-02274.

Numerical Quantifiers and Their Use in Reasoning with Negative Information*

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Amherst, New York 14226

Key words: Reasoning; logic based reasoning; quantifiers; numerical quantifiers; negative information; negation; representation of knowledge; formal representations; semantic networks.

Abstract

Numerical quantifiers provide simple means of formalizing such statements as, "at least three people are in that room", "at most fifteen people are in the elevator", and "everybody has exactly two parents". Although numerical quantifiers generalize the existential quantifier, they have different uses in reasoning. The existential quantifier is most useful for supplying referents for designating phrases with no previously explicitly mentioned referent. Numerical quantifiers are most useful for reasoning by the process of elimination. Numerical quantifiers would, therefore, be a useful addition to the operators of a reasoning program or deductive question-answering system. They have been added to SNePS, the Semantic Network Processing System, to further enhance its inference capabilities.

---

*This material is based on work supported in part by a Faculty Research Fellowship from the Research Foundation of State University of New York, and in part by the National Science Foundation under Grant No. MCS78-02274.

Introduction

Logic based reasoning programs, that is reasoning programs based on operators (connectives, quantifiers, modals) which have been studied as part of formal logical systems benefit from the fact that the inferential properties of their operators are clear and well known. They need not be restricted, however, to a minimal set of operators. Minimal sets of operators are useful for proving properties of logical systems such as consistency and completeness, but using a logical system for carrying out inferences is simplified (for people) by enlarging the set of basic operators. This is one reason that natural deduction systems like those of [Fitch, 1952], [Prawitz, 1965] and [Weyhrauch, 1977], with reasonable sets of connectives and two rules of inference for each one, are easier to use than axiomatic systems with minimal sets of connectives, rules and axioms.

This paper is motivated by an interest in programs that represent knowledge, including the knowledge of rules of reasoning, and that use those rules to perform reasoning. I believe that such programs are enhanced by the availability of a large set of operators that typify and formally model as many of the modes of human reasoning as possible. This paper discusses a set of operators, the numerical quantifiers, which can be implemented in reasoning programs as a single parameterized operator, and which model an important mode of human reasoning.

Numerical quantifiers, discussed briefly in [Tarski, 1965, pp. 63-64], are generalizations of the existential quantifier. They can be used to formalize such statements as:

There are at least two numbers z, such that $z+2<6$.

There are exactly two numbers x, such that $x^2+4=4x$.

There are at most two numbers y, such that $y+5<11-2y$.

[all from Tarski, 1965, pp. 64, 67].

One numerical quantifier is the more commonly encountered unique existential, expressed as $\exists!xA(x)$ in the notation of [Kleene, 1950, p. 199]. We will use the notation $\exists_i^j xA(x)$ for "there exists at least i and at most j x such that A(x)". The usual existential quantifier, $\exists xA(x)$, can then be considered an abbreviation of $\exists_1^\infty xA(x)$ (although later we will make a distinction), and the unique existential becomes $\exists_1^1 xA(x)$. In general, $\exists_n^n xA(x)$ are the "numerically definite quantifiers" mentioned in [Lemmon, 1978, pp. 165, 6].

We have found the numerical quantifiers particularly useful for the mode of reasoning by the process of elimination: if the maximal number of positive cases are found, the rest must be negative; if the maximal number of negative cases are found, the rest must be positive. Numerical quantifiers thus can introduce explicit negative information into a data base, and can make use of negative information to derive positive information. To set the stage for this discussion, we will first discuss the role of the simple existential quantifier in deductive question-answering.

The Existential Quantifier

Let us consider statements which:

1) include an existential quantifier;

2) are to be stored in the data base of a deductive question-answering system (QAS);

3) are to be used by the system to answer questions.

We will consider what contribution such statements (we call these, as well as other general statements, deduction rules) can make to the question-answering process.

Existential quantifiers can either be outside or inside the scope of universal quantifiers. If outside the scope of any universal quantifier, for example "There is a man who owns a dog" or $\exists x(Man(x)\&\exists y(Dog(y)\&Owns(x,y)))$, there is no need to retain the quantifier in the data base, one can simply create a new individual constants (Skolem constants) and substitute them for the quantified variables, storing the three facts Man(m1), Dog(d1), and Owns(m1,d1).

Existential quantifiers within the scope of universal quantifiers can be eliminated by replacing them with Skolem functions. So,

(1) "Every person has a mother"

can be represented by $\forall x(Person(x)\rightarrow\exists y(Person(y)\&Mother(y,x)))$ or by $\forall x(Person(x)\rightarrow(Person(f(x))\&Mother(f(x),x)))$, where f is a new function. This rule could be used to answer the question, "Does John have a mother?", but the point of the Skolem function is that for each person a new person must be postulated to be his or her mother. So, knowing just this rule, and that John is a person, if we asked "Who is John's mother?", the answer would be some individual about whom we know nothing except that

she is a person and is John's mother.

It may seem strange that asking a question can cause the creation of a new individual, but consider definite descriptions that refer to individuals which have not been explicitly introduced. This often arises in statements. Consider "The mother of John owns a dog", or "John's mother owns a dog". Normally, we would look in the data base for John's mother and assert that she owns a dog, but in this case there is no record of John's mother in the data base. However, rule (1) justifies creating a new individual to be John's mother. "John's mother owns a dog" presupposes that John has a mother. With neither an explicit mother, nor the rule, the sentence has a failed presupposition and should not be accepted. Compare the situation in which we ask, "Does John's mother own a dog?" In the absence of an explicit mother and rule (1), the correct response, as [Kaplan, 1978] points out, would be something like "I didn't know that John had a mother". In the presence of rule (1), however, the correct response would be, "I don't know". (Under the closed world assumption that any statement not in the data base is false [Reiter, 1978], the answers would be, "John doesn't have a mother", and "No", respectively).

What if the data base contained both rule (1) and

(2) Jane is John's mother

and we input, "John's mother owns a dog."? In this case, rule (1) is at best useless, and at worst, harmful. If the rule were activated, it would have to produce a new mother, and the phrase "John's mother" would be ambiguous. If rule (1) were

not activated, "Jane owns a dog" would be stored. This is technically wrong (consider replacing "mother" with "parent"), but probably what the speaker intended.

In summary, existential quantifiers need be stored in the data base of a QAS only when within the scope of a universal quantifier, and they are most useful for supplying referents for designating phrases with no previously explicitly mentioned referent.

Maximal Numerical Quantifiers

Let us consider the data base containing (2), "Jane is John's mother" and the question, "Is Mary John's mother?" In order to get the correct answer, "No", we need the rule

(3) Every person has at most one mother.

(We will ignore, in this paper, the problem of identity or extensional equivalence. That is, the even more correct answer, "Only if Mary is the same person as Jane". See [Shapiro, 1978] for a solution to this problem using a combination of path tracing and deduction rules.) Let us call quantifiers of the form $\exists^{j}xA(x)$, read "there exists at most j x such that A of x", maximal numerical quantifiers. In this notation, (3) becomes

(3') $\forall x(\text{Person}(x)\rightarrow\exists^{1}y(\text{Person}(y)\&\text{Mother}(y,x)))$

Another example of maximal numerical quantifiers is, "Every football team must have at most eleven players on the field at any time".

Unlike the simple existential quantifier, the maximal numerical quantifiers do not justify the introduction of new

individuals. However, we can derive negative statements from them once the maximal number of individuals satisfying the quantified statement are known. In our example, rule (3) justifies the answer, "No, Mary is not John's mother".

Suppose the data base consists of rules (1) and (3) only. Then if we ask, "Is Jane John's mother?", it might seem that rule (1) would create a new Skolem constant to be John's mother, and then by rule (3) the answer would be "No". However, rule (1) must not be invoked in this case, because it is illegal to instantiate an existentially quantified variable to a constant we already know something about. Since John's mother is not known explicitly, Jane is not ruled out and the correct answer in this case is "I don't know".

The formula $\exists^{j}xA(x)$ is therefore useful when we already know j different individuals satisfying A and are asked if a (j+1)st individual, t, also satisfies A. The formula ~A(t) is then derivable.

Minimal Numerical Quantifiers

Let us now consider formulas of the form,

(4) $\exists_i xA(x)$

read "There exists at least i x such that A of x". We will call quantifiers of this form, minimal numerical quantifiers. What useful information does (4) provide that is not provided by $\exists xA(x)$? If the universe under discussion contains n individuals, and we already know of n-i individuals y such that

~A(y), we can deduce about any (n-i+1)st individual, t, that A(t). Consider five faculty members, three of whom are in a meeting. If I know who the five people are, and I've seen two in the hall, I can deduce who is in the meeting.

Since the usefulness of minimal numerical quantifiers depends on some universe of objects, it is convenient to introduce domain restricted minimal numerical quantifiers. We will use the notation $\exists_i x(P(x):Q(x))$, where P(x) and Q(x) are arbitrary formulas with x free, to mean, "of all objects x such that P(x), at least i of them satisfy Q". In a data base without the closed world assumption, we can seldom be sure that the objects known to satisfy P are the only ones that actually do. For example, the following corpus, representing the example above, is insufficient for deducing who is in the meeting.

(5) $\exists_3 x(MEMBER(x,FACULTY):IN(x,MEETING))$

(6) $\forall x(IN(x,HALL) \rightarrow \sim IN(x,MEETING))$

(7) MEMBER(PAT,FACULTY)

(8) MEMBER(GABOR,FACULTY)

(9) MEMBER(NICK,FACULTY)

(10) MEMBER(JOHN,FACULTY)

(11) MEMBER(STU,FACULTY)

(12) IN(PAT,HALL)

(13) IN(NICK,HALL)

However, it may be that whoever provided rule (5) knows that there are only five faculty members. To record such information, we

will add another parameter to the minimal numerical quantifiers giving the schema, ${}_n\exists_i x(P(x):Q(x))$, where n is the number of objects which satisfy P. Notice that this amounts to a closed sub-world assumption. If we replace (5) in the above corpus by

(5') ${}_5\exists_3 x(\text{MEMBER}(x,\text{FACULTY}):\text{IN}(x,\text{MEETING}))$

stating that "Of the five faculty members, at least three are in the meeting", then we can derive IN(GABOR,MEETING), IN(JOHN,MEETING) and IN(STU,MEETING).

With minimal numerical quantifiers, negative information can be used to deduce positive information. Given the rule ${}_n\exists_i x(P(x):Q(x))$, and $n-i$ individuals y such that $P(y)\&\sim Q(y)$, we can deduce for any other individual t such that $P(t)$ holds that $Q(t)$ also holds.

Numerical Quantifiers

The minimal and maximal numerical quantifiers can be combined into what we shall simply call the numerical quantifiers

(14) ${}_n\exists_i^j x(P(x):Q(x))$

which are read, "Of the n individuals x such that $P(x)$, at least i and at most j are such that $Q(x)$", or simply, "Between i and j of the n Ps are Qs". The other quantifiers we have discussed may be obtained by leaving out appropriate parts of schema (14).

The regular existential quantifier, $\exists x A(x)$, is the same as the numerical quantifier

(15) $\exists_1^\infty x A(x)$.

It is now clear why it seldom helps us determine whether A(t) holds for a fixed individual, t. Rule (15) cannot derive ~A(t), since there is no maximum -- all individuals might satisfy A. Rule (15) can seldom produce A(t), since that would require knowing that all other individuals satisfy ~A, and we rarely have a finite list of all the individuals in the domain. Because of this, the implementor of a deductive question-answering system may wish to distinguish the existential quantifier from the numerical quantifiers, and continue to prohibit invokation of a rule that would bind an existentially quantified variable to a constant.

Numerical Quantifiers in SNePS

Numerical quantifiers have recently been added to SNePS, the Semantic Network Processing System [Shapiro, 1979], which already included universal and existential quantifiers as well as a set of non-standard connectives (see also [Shapiro, 1977]). SNePS accepts numerically quantified formulas of the form

(16) $_{n}\exists_{i}^{j}\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$

(17) $_{n}\exists_{i}\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$

(18) $\exists^{j}\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$

where $k\geq 0$ and $\bar{x}$ represents a sequence of variables each of which is free in at least one of $P_1(\bar{x}),\ldots,P_k(\bar{x}),Q(\bar{x})$. The meaning of (16) is that of the n combinations of individuals satisfying $P_1(\bar{x})\&\ldots\&P_k(\bar{x})$, at least i and at most j of these combinations also satisfy $Q(\bar{x})$. Rule (17) means that at least i of the n combinations of individuals satisfying

$P_1(\bar{x})\&...\&P_k(\bar{x})$ also satisfy $Q(\bar{x})$. Rule (18) states that at most j of the combinations of individuals satisfying $P_1(\bar{x})\&...\&P_k(\bar{x})$ also satisfy $Q(\bar{x})$.

As an example of (16) involving a sequence of variables, consider the many-many relationship of dog ownership. Several people in a family may own one dog, and several dogs may be owned by the same person. The formula

$$_5\exists^4_2 x,y(\text{Person}(x),\text{Dog}(y),\text{Owns}(x,y):\text{Spoils}(x,y))$$

says that of the **five dog ownership relations** (which may involve one to five dogs and one to five people), between two and four involve spoiling the dog.

Rules of the form of (16) are represented in the SNePS network by a node with:

an auxiliary arc labeled EMIN to i

an auxiliary arc labeled EMAX to j

an auxiliary arc labeled ETOT to n

descending arcs labeled PEVB to the nodes representing the variables in $\bar{x}$

descending arcs labeled &ANT to the nodes representing the formulas $P_1(\bar{x}),...,P_k(\bar{x})$

a descending arc labeled CQ to the node representing $Q(\bar{x})$.

In SNePS, auxiliary arcs may connect semantic network nodes to arbitrary data structures, including numbers. A descending arc goes from a network node to another network node and has a paired ascending arc in the reverse direction.

If a rule of the form of (16) is in the network, and a derivation of $Q(\bar{a})$ is requested, where $\bar{a}$ is a substitution instance of $\bar{x}$, subgoal derivations of $P_1(\bar{x}),\ldots,P_k(\bar{x})$, and $Q(\bar{x})$ are begun. One of the following cases will occur:

1. j substitution instances of $\bar{x}$ are found for which $P_1(\bar{x})\&\ldots\&P_k(\bar{x})\&Q(\bar{x})$. If this occurs, let $Q'(\bar{x})$ be $\sim Q(\bar{x})$.

2. n-i substitution instances of $\bar{x}$ are found for which $P_1(\bar{x})\&\ldots\&P_k(\bar{x})\&\sim Q(\bar{x})$. If this occurs, let $Q'(\bar{x})$ be $Q(\bar{x})$.

3. Neither case (1) nor case (2) ever occurs. In this case rule (16) is incapabile, in the current data base, of deriving either $Q(\bar{a})$ or $\sim Q(\bar{a})$

If case (1) or (2) occurs, there are two possibilities. If $P_1(\bar{a})\&\ldots\&P_k(\bar{a})$ has been derived, $Q'(\bar{a})$ is thereby derived. If this has not happened, the process implementing this rule changes itself into one implementing the rule $P_1(\bar{a})\&\ldots\&P_k(\bar{a})\rightarrow Q'(a)$, retains the relevant data already accumulated, and continues processing.

This has been a simplified account since the details would require a general discussion of the way SNePS processes deduction rules. This general discussion is contained in [Shapiro and McKay, forthcoming]. A few points, however, are worth noting. A process which implements a deduction rule, i.e.

uses the rule for a specific derivation, creates a process for each subgoal deduction. All processes are executed in parallel, so the occurrence of case (3) above, would not prevent some other rule's derivation of $Q(\bar{a})$ or $\sim Q(\bar{a})$. A process that derives many substitution instances of a formula returns them as they are generated rather than waiting for completion. This is why the two possibilities mentioned above can occur.

Examples

In this section, we will show SNePS runs of some of the examples from above. Lines beginning with "**" or "*" were typed by the user. The character ";" indicates that the rest of the line is a comment. Each example was begun with an empty network so that the two "mother" examples would not interact. Input is in SNePSUL, the SNePS User Language, [Shapiro, 1979], but it is hoped that, with the aid of the comments and the previous discussion, the reader will be able to follow it. The runs were transcribed to make them easier to read, and edited only to add the comments in the right margin and to remove typographical errors and some trace printing. SNePS is written in Lisp and runs interactively on a CYBER 173. A compiled version of SNePS was used for the first two examples.

Example 1:

```
**;Every person has a mother.
*(BUILD AVB $X                                              ;  ∀x
*       ANT (BUILD MEMBER *X CLASS PERSON)                  ;   [Person(x)→
*        CQ (BUILD EVB $Y MIN 2 MAX 2                       ;    ∃y
*              ARG ((BUILD MEMBER *Y CLASS PERSON)          ;    (Person(y)&
*                   (BUILD A1 *Y R MOTHER A2 *X)))          ;     Mother(y,x))]
(M5); This is the SNePS node representing the rule.
62 MSECS


**;John is a person.
*(BUILD MEMBER JOHN CLASS PERSON)
(M6)
7 MSECS


**;John's mother owns a dog.
*(DESCRIBE
*(BUILD A1 (FIND A1- (DEDUCE A1 %X R MOTHER A2 JOHN));  John's mother
*       R OWNER                                      ;  owns
*       A2 (BUILD MEMBER- (BUILD CLASS DOG))))       ;  a dog.
(M12 (A1 (B1)) (R (OWNER)) (A2 (M11))); The assertion that was built.
(DUMPED)
266 MSECS


**(DUMP B1 M11); To find out everything about B1 and M11.
(B1 (MEMBER- (M9)) (A1- (M12 M8)))
(M11 (A2- (M12)) (MEMBER- (M1Ø)))
(DUMPED)
1Ø MSECS


**(DUMP M9 M12 M8 M1Ø); To print all assertions about B1 and M11.
(M9 (MEMBER (B1)) (CLASS (PERSON)))      ;  B1 is a person.
(M12 (A1 (B1)) (R (OWNER)) (A2 (M11)))   ;  B1 owns M11.
(M8 (A1 (B1)) (R (MOTHER)) (A2 (JOHN)))  ;  B1 is John's mother.
(M1Ø (MEMBER (M11)) (CLASS (DOG)))       ;  M11 is a dog.
(DUMPED)
17 MSECS
```

Example 2:

```
**;Jane is a person.
*(BUILD MEMBER JANE CLASS PERSON)
(M13)
8 MSECS
```

```
**;John is a person.
*(BUILD MEMBER JOHN CLASS PERSON)
(M14)
8 MSECS


**;Jane is John's mother.
*(BUILD A1 JANE R MOTHER A2 JOHN)
(M15)
1Ø MSECS


**;Every person has at most one mother.
*(BUILD AVB $X                                                  ;  ∀x
*       ANT (BUILD MEMBER *X CLASS PERSON)                      ;    [Person(x)→
*        CQ (BUILD EMAX 1 PEVB $Y                               ;       ∃¹y
*                  &ANT (BUILD MEMBER *Y CLASS PERSON)          ;          (Person(y):
*                    CQ (BUILD A1 *Y R MOTHER A2 *X)))          ;           Mother(y,x))]
(M21)
62 MSECS


**;Mary is a person.
*(BUILD MEMBER MARY CLASS PERSON)
(M22)
7 MSECS


**;Is Mary John's mother?
*(DESCRIBE (DEDUCE A1 MARY R MOTHER A2 JOHN))
(M24 (MIN (Ø)) (MAX (Ø)) (ARG (M23)))       ;  It is not the case that
(M23 (A1 (MARY)) (R MOTHER)) (A2 (JOHN)))   ;     Mary is John's mother.
(DUMPED)
827 MSECS


Example 3:


**;At least 3 of the 5 faculty members are in the meeting.
*(BUILD ETOT 5 EMIN 3 PEVB $X                   ;  5∃3x
*       &ANT (BUILD MEMBER *X CLASS FACULTY)    ;     [MEMBER(x,FACULTY)
*         CQ (BUILD A1 *X R IN A2 MEETING))     ;        →IN(x,MEETING)]
(M4)
695 MSECS


**;Whoever is in the hall is not in the meeting.
*(BUILD AVB $X                                                  ; ∀x
*       ANT (BUILD A1 *X R IN A2 HALL)                          ;  [IN(x,HALL)
*        CQ (BUILD MIN Ø MAX Ø                                  ;     →~
*                  ARG (BUILD A1 *X R IN A2 MEETING)))          ;      IN(x,MEETING)]
(M8)
193 MSECS
```

```
**;Pat is a faculty member.
*(BUILD MEMBER PAT CLASS FACULTY)
(M9)
3Ø MSECS


**;Gabor is a faculty member.
*(BUILD MEMBER GABOR CLASS FACULTY)
(M1Ø)
33 MSECS


**;Nick is a faculty member.
*(BUILD MEMBER NICK CLASS FACULTY)
(M11)
34 MSECS


**;John is a faculty member.
*(BUILD MEMBER JOHN CLASS FACULTY)
(M12)
31 MSECS


**;Stu is a faculty member.
*(BUILD MEMBER STU CLASS FACULTY)
(M13)
33 MSECS


**;Pat is in the hall.
*(BUILD A1 PAT R IN A2 HALL)
(M14)
4Ø MSECS


**;Nick is in the hall.
*(BUILD A1 NICK R IN A2 HALL)
(M15)
42 MSECS


**;Who is at the meeting?
*(DESCRIBE (DEDUCE A1 %X R IN A2 MEETING))
(M17 (MIN (Ø)) (MAX (Ø)) (ARG (M16)))        ; It is not the case that
(M16 (A1 (PAT)) (R (IN)) (A2 (MEETING)))     ;    Pat is in the meeting.
(M19 (MIN (Ø)) (MAX (Ø)) (ARG (M18)))        ; It is not the case that
(M18 (A1 (NICK)) (R (IN)) (A2 MEETING)))     ;    Nick is in the meeting.
(M20 (A1 (GABOR)) (R (IN)) (A2 (MEETING))) ; Gabor is in the meeting.
(M21 (A1 (JOHN)) (R (IN)) (A2 (MEETING)))  ; John is in the meeting.
(M22 (A1 (STU)) (R (IN)) (A2 (MEETING)))   ; Stu is in the meeting.
(DUMPED)
4Ø33 MSECS
```

Example 4:

```
**;Of 5 dog ownership relationships between 2 and 4 involve spoiling.
*(BUILD ETOT 5 EMIN 2 EMAX 4 PEVB ($X $Y)     ; 5∃[2,4]x,y
*       &ANT ((BUILD MEMBER *X CLASS PERSON) ;     [MEMBER(x,PERSON)
*             (BUILD MEMBER *Y CLASS DOG)    ;      &MEMBER(y,DOG)
*             (BUILD A1 *X R OWNER A2 *Y))   ;      &OWNS(x,y)
*         CQ (BUILD A1 *X R SPOILS A2 *Y))   ;      →SPOILS(x,y)]
(M5)
284 MSECS


**;John is a person.
*(BUILD MEMBER JOHN CLASS PERSON)
(M6)
31 MSECS


**;Jane is a person.
*(BUILD MEMBER JANE CLASS PERSON)
(M7)
31 MSECS


**;Mary is a person.
*(BUILD MEMBER MARY CLASS PERSON)
(M8)
567 MSECS


**;Jim is a person.
*(BUILD MEMBER JIM CLASS PERSON)
(M9)
31 MSECS


**;Rover is a dog.
*(BUILD MEMBER ROVER CLASS DOG)
(M1Ø)
32 MSECS


**;Spot is a dog.
*(BUILD MEMBER SPOT CLASS DOG)
(M11)
30 MSECS


**;Lassie is a dog.
*(BUILD MEMBER LASSIE CLASS DOG)
(M12)
32 MSECS
```

```
**;John owns Rover.
*(BUILD A1 JOHN R OWNER A2 ROVER)
(M13)
41 MSECS

**;John owns Spot.
*(BUILD A1 JOHN R OWNER A2 SPOT)
(M14)
42 MSECS

**;Mary owns Lassie.
*(BUILD A1 MARY R OWNER A2 LASSIE)
(M15)
42 MSECS

**;Jane owns Spot.
*(BUILD A1 JANE R OWNER A2 SPOT)
(M16)
43 MSECS

**;Jim owns Lassie.
*(BUILD A1 JIM R OWNER A2 LASSIE)
(M17)
41 MSECS

**;John spoils Rover.
*(BUILD A1 JOHN R SPOILS A2 ROVER)
(M18)
44 MSECS

**;John spoils Spot.
*(BUILD A1 JOHN R SPOILS A2 SPOT)
(M19)
44 MSECS

**;Jane spoils Spot.
*(BUILD A1 JANE R SPOILS A2 SPOT)
(M20)
43 MSECS

**;Mary spoils Lassie.
*(BUILD A1 MARY R SPOILS A2 LASSIE)
(M21)
41 MSECS
```

```
**;Who spoils whom?
*(DESCRIBE (DEDUCE A1 %X R SPOILS A2 %Y))
(M18 (A1 (JOHN)) (R (SPOILS)) (A2 (ROVER)))   ; John spoils Rover.
(M19 (A1 (JOHN)) (R (SPOILS)) (A2 (SPOT)))    ; John spoils Spot.
(M20 (A1 (JANE)) (R (SPOILS)) (A2 (SPOT)))    ; Jane spoils Spot.
(M21 (A1 (MARY)) (R (SPOILS)) (A2 (LASSIE))) ; Mary spoils Lassie.
(M23 (MIN (Ø)) (MAX (Ø)) (ARG (M22)))         ; It is not the case that
(M22 (A1 (JIM)) (R (SPOILS)) (A2 (LASSIE)))   ;   Jim spoils Lassie.
(DUMPED)
3965 MSECS
```


Summary

We have discussed the roles of existential and numerical quantifiers in reasoning programs. The roles are different and both are important. The existential quantifier is most useful for supplying referents for designating phrases with no previously explicitly mentioned referent. Numerically quantified rules are concise representations of rules that govern reasoning by the process of elimination and thereby can introduce explicit negatives into a data base or can use negative statements for deriving positive statements. The most general schema for numerical quantifiers that we have discussed is ${}_n\exists{}_i^j\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$, which says that at least i and at most j of the n combinations of individuals that satisfy $P_1(\bar{x})\&\ldots\&P_k(\bar{x})$ also satisfy $Q(\bar{x})$. We showed how rules of this form can be represented in SNePS, the Semantic Network Processing System, and gave examples of SNePS runs that used such rules for carrying out inferences.

Acknowledgements

The author is grateful to Don McKay for help in SNePS development and to him and Rich Fritzson for SNePS and Lisp system support. He is also grateful to Brian Funt and Don McKay for comments on an earlier draft.

References

Fitch, F.B. Symbolic Logic: An Introduction. Ronald Press Co., New York, 1952.

Kaplan, S.J. Indirect responses to loaded questions. In D. Waltz, ed. TINLAP-2: Theoretical Issues in Natural Language Processing-2. ACM, New York, 1978, 202-209.

Kleene, S.C. Introduction to Metamathematics. D. Van Nostrand, Princeton, New Jersey, 1950.

Lemmon, E.J. Beginning Logic. Hackett, Indianapolis, 1978.

Prawitz, D. Natural Deduction - A Proof-Theoretical Study. Almqvist and Wiksell, Stockholm, 1965.

Reiter, R. On closed world data bases. In H. Gallaire and J. Minker, eds. Logic and Data Bases, Plenum Press, New York, 1978.

Shapiro, S.C. Representing and locating deduction rules in a semantic network. Proc. Workshop on Pattern-Directed Inference Systems. SIGART Newsletter, 63 (June, 1977), 14-18.

Shapiro, S.C. Path-based and node-based inference in semantic networks. In D. Waltz, ed. TINLAP-2: Theoretical Issues in Natural Language Processing-2. ACM, New York, 1978, 219-225.

Shapiro, S.C. The SNePS semantic network processing system. In N. Findler, ed. Associative Networks - The Representation and Use of Knowledge by Computers, Academic Press, New York, 1979, 179-203.

Shapiro, S.C. and McKay, D.P. The representation and use of deduction rules in a semantic network. Department of Computer Science, SUNY/Buffalo, Amherst, New York, forthcoming.

Tarski, A. Introduction to Logic and to the Methodology of Deductive Sciences. Oxford University Press, New York, 1965.

Weyhrauch, R.W. A users manual for FOL, Memo AIM-235.1, Stanford Artificial Intelligence Laboratory, Stanford, California, 1977.