NUMERICAL QUANTIFIERS AND THEIR USE IN REASONING WITH NEGATIVE INFORMATION

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
4226 Ridge Lea Road
Amherst, New York 14226

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.

1. 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 [l] , [4] and [11], 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.

*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, discussed briefly in [10, 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 [10, pp. 64, 67].)

One numerical quantifier is the more commonly encountered unique existential, expressed as $\exists!xA(x)$ in the notation of [2, 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 [3, 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.

2. 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 $\exists x(\mathrm{Man}(x) \& \exists y(\mathrm{Dog}(y) \& \mathrm{Owns}(x,y)))$, there is no need to retain the quantifier in the data base, one can simply create new individual constants (Skolem constants) and substitute them for the quantified variables, storing, in this example, the three facts Man(ml), Dog(dl), and Owns(ml,dl).

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 $\forall x(\mathrm{Person}(x) \rightarrow \exists y(\mathrm{Person}(y) \& \mathrm{Mother}(y,x)))$ or by $\forall x(\mathrm{Person}(x) \rightarrow (\mathrm{Person}(f(x)) \& \mathrm{Mother}(f(x),x)))$, 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 (1), 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, as in the statement "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.

If the data base contained both rule (1) and "Jane is John's mother", and we input, "John's mother owns a dog," 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.

3. MAXIMAL NUMERICAL QUANTIFIERS

Consider the data base containing "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
(2) 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 [7] for a solution to this problem using a combination of path tracing and deduction rules.) Let us call quantifiers of the form $\text{Ǝ}^j xA(x)$, read "there exists at most j x such that A of x", maximal numerical quantifiers. In this notation, (2) becomes
(2') $\forall x(\mathrm{Person}(x) \rightarrow \text{Ǝ}^1 y(\mathrm{Person}(y) \& \mathrm{Mother}(y,x)))$

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 (2) justifies the answer, "No, Mary is not John's mother".

If a data base consisted of rules (1) and (2) only, and we asked, "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 (2) 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 $\sim A(t)$ is then derivable.

4. MINIMAL NUMERICAL QUANTIFIERS

Now consider formulas of the form, $\exists_i xA(x)$, read "There exists at least i x such that A of of x". We will call quantifiers of this form, minimal numerical quantifiers. What useful information does this 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 $\sim A(y)$, we can deduce about any (n-i+1)st individual, t, that A(t). Consider five professors, 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 [5], 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.

(3) $\exists_3 x(\text{Professor}(x):\text{In}(x,\text{meeting}))$
(4) $\forall x(\text{In}(x,\text{hall})\rightarrow\sim\text{In}(x,\text{meeting}))$
(5) Professor(Pat)
(6) Professor(Gabor)
(7) Professor(Nick)
(8) Professor(John)
(9) Professor(Stu)
(10) In(Pat,hall)
(11) In(Nick,hall)

However, it may be that whoever provided rule (3) knows that there are only five professors. 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 (3) in the above corpus by

(3') ${}_5\exists_3 x(\text{Professor}(x):\text{In}(x,\text{meeting}))$

stating that "Of the five professors, 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.

5. NUMERICAL QUANTIFIERS

The minimal and maximal numerical quantifiers can be combined into what we shall simply call the numerical quantifiers, ${}_n\exists_i^j x(P(x):Q(x))$, 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 this schema.

If a data base contains a rule of the form ${}_n\exists_i^j x(P(x):Q(x))$ and j individuals are found satisfying P(x)&Q(x), it may be deduced that the remaining n-j individuals satisfying P(x) satisfy $\sim Q(x)$. However, if n-i individuals are found to satisfy $P(x)\&\sim Q(x)$, it may be deduced that the remaining i individuals satisfying P(x) also satisfy Q(x).

The regular existential quantifier, $\exists xA(x)$, is the same as the numerical quantifier $\exists_1^\infty xA(x)$. It is now clear why it seldom helps us determine whether A(t) holds for a fixed individual, t. $\exists_1^\infty xA(x)$ cannot derive $\sim A(t)$, since there is no maximum -- all individuals might satisfy A. It can seldom produce A(t), since that would require knowing that all other individuals satisfy $\sim A$, and we rarely have a finite list of all the individuals in the domain. Because of this, the implementor of a deductive QAS may wish to distinguish the existential quantifier from the numerical quantifiers, and continue to prohibit invocation of a rule that would bind an existentially quantified variable to a constant.

6. NUMERICAL QUANTIFIERS IN SNePS

Numerical quantifiers have recently been added to SNePS, the Semantic Network Processing System [6; 8], which already included universal and existential quantifiers as well as a set of non-standard connectives. SNePS accepts numerically quantified formulas of the form

(12) ${}_n\exists_1^j\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$
(13) ${}_n\exists_1\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$
(14) $\exists^j\bar{x}(P_1(\bar{x}),\ldots,P_k(\bar{x}):Q(\bar{x}))$

where $k\geq 0$ and $\bar{x}$ represents a sequence of variables $<x_1,\ldots,x_m>$ 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 (12) is that there are n sequences of individuals, $<t_1,\ldots,t_m>$ such that $(P_1(\bar{x})\&\ldots\&P_k(\bar{x}))[t_1/x_1,\ldots,t_m/x_m]$ is true and that at least i and at most j of these n sequences also satisfy $Q(\bar{x})$. The meanings of (13) and (14) are the appropriate simplifications of this.

As an example of (12), consider the many-many relationship of dog ownership. Several people may jointly own one dog, and several dogs may be owned by the same person. The formula $_5\exists_2^4 x,y(\text{Person}(x),\text{Dog}(y),\text{Owns}(x,y):\text{Spoils}(x,y))$ says that of the five dog ownership relations (e.g. <John,Rover>, <John,Spot>, <Jane,Spot>, <Mary, Lassie>, and <Jim,Lassie>), between two and four involve spoiling the dog.

Rules of the form of (12) 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 $P_1(\bar{x}),\ldots,P_k(\bar{x})$; 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.

7. EXAMPLES

In this section, we will show SNePS runs of the three major numerical quantifier examples from above. Lines beginning with "**" or "*" were typed by the user. The character ";" indicates that the rest of the line is a comment. Input is in SNePSUL, the SNePS User Language, [8]. Each input and each deduced fact is commented by its English translation. Each rule is also commented by its translation into the logical syntax used previously in this paper. The runs were transcribed to make them easier to read, and edited only to add the comments and to remove some trace printing. SNePS is written in Lisp and runs interactively on a CYBER 173. A compiled version was used for these examples.

7.1 Example 1

We assert that Jane is John's mother and that everyone has at most one mother. Then we ask if Mary is John's mother, and SNePS responds that she is not.

```
**;Jane is a person.
*(BUILD MEM JANE CLASS PERSON)
(M13) ;The SNePS node representing the assertion.
8 MSECS
**;John is a person.
*(BUILD MEM JOHN CLASS PERSON)
(M14)
8 MSECS
**;Jane is John's mother.
*(BUILD A JANE R MOTHER O JOHN)
(M15)
12 MSECS
**;Every person has at most one mother.
*;∀x[Person(x)→ ∃^1y(Person(y) : Mother(y,x))]
*(BUILD AVB $X
* ANT (BUILD MEM *X CLASS PERSON)
*  CQ (BUILD EMAX 1 PEVB $Y
*      &ANT (BUILD MEM *Y CLASS PERSON)
*         CQ (BUILD A *Y R MOTHER O *X)))
(M21)
65 MSECS
**;Mary is a person.
*(BUILD MEM MARY CLASS PERSON)
(M22)
7 MSECS
**;Is Mary John's mother?
* (DESCRIBE (DEDUCE A MARY R MOTHER O JOHN))
(M24 (MIN(0)) (MAX(0)) (ARG(M23)))
     ; It is not the case that
(M23 (A(MARY)) (R(MOTHER)) (O(JOHN)))
     ; Mary is the mother of John.
(DUMPED)
1120 MSECS
```

7.2 Example 2

At least three of the five professors are in the meeting. We see Pat and Nick in the hall. When we ask who is in the meeting, SNePS deduces that Pat and Nick are not, but Gabor, John, and Stu are.

```
**;At least 3 of 5 professors are in a meeting.
*;5∃3x[Professor(x)→In(x,meeting)]
*(BUILD ETOT 5 EMIN 3 PEVB $X
* &ANT (BUILD MEM *X CLASS PROFESSOR)
*   CQ (BUILD A *X R IN O MEETING))
(M4)
44 MSECS
**;Whoever is in the hall is not in the meeting.
*;∀x[In(x,hall)→ ~In(x,meeting)]
*(BUILD AVB $X
* ANT (BUILD A *X R IN O HALL)
*  CQ (BUILD MIN 0 MAX 0
*      ARG (BUILD A *X R IN O MEETING)))
(M8)
50 MSECS
**;Pat is a professor.
*(BUILD MEM PAT CLASS PROFESSOR)
(M9)
10 MSECS
**;Gabor is a professor.
*(BUILD MEM GABOR CLASS PROFESSOR)
(M10)
8 MSECS
**;Nick is a professor.
*(BUILD MEM NICK CLASS PROFESSOR)
(M11)
8 MSECS
**;John is a professor.
*(BUILD MEM JOHN CLASS PROFESSOR)
(M12)
8 MSECS
**;Stu is a professor.
*(BUILD MEM STU CLASS PROFESSOR)
```

```
(M13)
9 MSECS
**;Pat is in the hall.
*(BUILD A PAT R IN 0 HALL)
(M14)
11 MSECS
**;Nick is in the hall.
*(BUILD A NICK R IN 0 HALL)
(M15)
10 MSECS
**;Who is in the meeting?
*(DESCRIBE (DEDUCE A %X R IN 0 MEETING))
(M17 (MIN(O)) (MAX(O)) (ARG(M16)))
(M16 (A(PAT)) (R(IN)) (O(MEETING)))
      ; Pat is not in the meeting.
(M19 (MIN(O)) (MAX(O)) (ARG(M18)))
(MI8 (A(NICK)) (R(IN)) (0(MEETING)))
      ; Nick is not in the meeting.
(M20 (A(GABOR)) (R(IN)) (O(MEETING)))
      ; Gabor is in the meeting.
(M21 (A(JOHN)) (R(IN)) (0(MEETING)))
      ; John is in the meeting.
(M22 (A(STU)) (R(IN)) (O(MEETING)))
      ; Stu is in the meeting.
(DUMPED)
1427 MSECS
```

7 - .3 Example 2

We assert that between two and four dog owner inp relations involve spoiling, and assert four such spoiling relations. SNePS deduces that Jim does not spoil Lassie.

```
**;Of 5 dog ownership relations,
*  ,between 2 and 4 involve spoiling.
* , 532Xy member(x,person),Member(y,dog) ,
         Owns(x,y) : Spoils(x,y)]
•* (BUILD ETOT 5 EMIN 2 EMAX 4 PEVB($X $Y)
*  6ANT ((BUILD MEM *X CLASS PERSON)
*        (BUILD MEM *Y CLASS DOG)
*        (BUILD A *X R OWNS 0 *Y))
    CQ (BUILD A *X R SPOILS 0 *Y))
(M5)
81 MSECS
**;John is a person.
*(BUILD MEM JOHN CLASS PERSON)
(M6)
10 MSECS
**;Jane is a person.
*(BUILD MEM JANE CLASS PERSON)
(M7)
10 MSECS
**;Mary is a person.
*(BUILD MEM MARY CLASS PERSON)
(M8)
9 MSECS
**;Jim is a person.
*(BUILD MEM JIM CLASS PERSON)
(M9)
9 MSECS
**;Rover is a dog.
*(BUILD MEM ROVER CLASS DOG
(M10)
9 MSECS
**;Spot is a dog.
*(BUILD MEM SPOT CLASS DOG)
(M11)
12 MSECS
**;Lassie is a dog
*(BUILD MEM LASSIE CLASS DOG)
(M12)
10 MSECS
**;John owns Rover.
*(BUILD A JOHN R OWNS 0 ROVER)
(M13)
11 MSECS
**;John owns Spot.
*(BUILD A JOHN R OWNS 0 SPOT)
(M14)
11 MSECS
**;Mary owns Lassie.
*(BUILD A MARY R OWNS 0 LASSIE)
(M15)
13 MSECS
**;Jane owns Spot.
*(BUILD A JANE R OWNS 0 SPOT)
(M16)
11 MSECS
**;Jim owns Lassie.
*(BUILD A JIM R OWNS 0 LASSIE)
(M17)
12 MSECS
**;John spoils Rover.
*(BUILD A JOHN R SPOILS 0 ROVER)
(M18)
10 MSECS
**;John spoils Spot.
*(BUILD A JOHN R SPOILS 0 SPOT)
(M19)
12 MSECS
**;jane spoils Spot.
*(BUILD A JANE R SPOILS 0 SPOT)
(M20)
12 MSECS
**;Mary spoils Lassie.
*(BUILD A MARY R SPOILS 0 LASSIE)
(M21)
11 MSECS
**;Who spoils whom?
*(DESCRIBE (DEDUCE A %X R SPOILS 0 %Y))
(M18 vA(JOHN)) (R(SPOILS)) (0(ROVER)))
      ;John spoils Rover.
(M19 (A(JOHN)) (R(SPOILS)) (O(SPOT)))
      ;John spoils Spot.
(M20 (A(JANE)) (R(SPOILS)) (O(SPOT)))
      ;Jane spoils Spot.
(M21 (A(MARY)) (R(SPOILS)) (O(LASSIE)))
      ;Mary spoils Lassie.
(M23 (MIN(O)) (MAX(O)) (ARG(M22)))
(M22 (A(JIM)) (R(SPOILS)) (O(LASSIE)))
      ;Jim does not spoil Lassie.
(DUMPED)
1341 MSECS
```

8. 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 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 sequences of individuals that satisfy $P_1(\bar{x})\&\ldots\&P_k(\bar{x})$ also satisfy Q(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.

Two aspects of numerical quantifiers might limit their immediate usefulness. One is the n parameter, required whenever the minimal parameter is present. In formulating numerically quantified statements, we have found specifying n to be bothersome. The parameter could be eliminated if the inference system had another means of determining how many individuals satisfy the restriction, for example if the closed world assumption held. The other problem is that inherent in any counting argument is the assumption that any two individuals are, in fact, distinct. If this assumption does not hold, the use of numerical quantifiers depends on a solution to the identity problem mentioned in Sec. 3. Except for these two problems, numerical quantifiers seem to be useful additions to the tool kit of representation and inference.

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

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

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

[3] Lemmon, E.J. Beginning Logic. Hackett, Indianapolis, 1978.

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

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

[6] 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.

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

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

[9] 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.

[10] Tarski, A. Introduction to Logic and to th Methodology of Deductive Sciences. Oxford University Press, New York, 1965.

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