Languages with Self-Reference I: Foundations

(or--We can have everything in first-order logic!)

Donald Perlis
University of Maryland
College Park, Maryland 20742

Abstract

It is argued that a proper treatment of cognitive notions such as beliefs and concepts should allow broad and consistent expression of syntax and semantics, and that this in turn depends on self-reference. A theory of quotation and unquotation is presented to this end that appears to make unnecessary the usual hierarchical and non-first-order constructions for these notions. In the current paper (part I) the underlying theory is presented; a sequel will treat in more detail the applications to cognition.

1. Introduction

Language provides a kind of discrete representation of reality. This serves the purpose of facilitating planning by providing discrete computable representations (statements) of what is possible in a given domain. So viewed, statements are part of the very world they purport to describe. It is our contention that a proper understanding of language will in the long run be found to depend upon this kind of self-referential ability. Herein is found the main theme of this paper: the interplay of assertion and meaning, syntax and truth.

All syntactic features then should be themselves expressible in quoted form, allowing the user to inspect and comment on usage. "The user" here means the reasoning system, be it a human, robot, or other. The kind of flexible system envisaged is one that will inevitably make errors due to the complexity of its environment, and that will then have on occasion to

re-evaluate its representations. This is the reason that we insist on a language that has expressions naming its own expressions: it will be crucial to be able to isolate a statement as such, and state that it is in error, and even to point out the exact expressions that should be changed.

It is worth dwelling on this further. Some proposals (e.g., McDermott and Doyle [1]) simply view old assertions to have been taken away or deleted, rather than noting them to be imperfect. That is, the old assertion is simply gone altogether by definition in the new state of the reasoning mechanism. As an example, the conclusion that Tweety can fly, based on the information that Tweety is a bird and on the absence of information that she cannot fly, disappears on the introduction of new information to the effect that Tweety cannot fly, with no trace of the former conclusion to the contrary. But then the fact of an error having been made and then corrected is not itself represented and so is not something that can be reasoned about with these same mechanisms. This leads to the usual complaint that process information is not cleanly represented in the system itself. We seek here to devise a language closer in spirit to natural language in that the language can be reasoned about in that same language itself.

As a consequence, a principal concern of this paper is the need to refer to certain statements as true (and false). The notion of truth appears a crucial one for any treatment of information that is utilized inside a reasoning system whose own behavior is itself to be reasoned about. This is easy to see: even though we may not *know* a statement of ours to be true, we do want to be able on certain occasions to

consider that it may be so (or that it may not). Without such a capability, error correcting, even of the deletion sort, cannot be done. What is being urged here, is to let such judgements come out in the open, so that the reasoning apparatus can be brought to bear on them.

As an example of the kind of reasoning we have in mind, consider a reasoner R that has belief B, then later learns not-B, recalls having believed B, and concludes that the statement that all its beliefs are true was itself false and therefore may well still be false. This involves temporal and inductive inference as well as a number of other things. We do not address all the aspects of this problem here; it does however illustrate the importance of the roles of truth and self-reference in commonsense reasoning. (See Perlis [2] for a broader discussion of the various aspects of the kind of reasoning just mentioned.)

In this part I of our investigation, we concentrate attention on the first-order truth definitional issues; more detailed analyses of cognitive notions and difficulties with modal and other treatments in the literature will be given in a sequel. Suffice it to say here that these approaches do not allow for unrestricted reference to syntax, and so do not satisfy our expressive criteria.

2. Some History

Let us start by reviewing some history. Gottlob Frege developed the first formal quantificational logic over a period of more than two decades

culminating in 1903. This consisted of a precise syntax, a set of inference rules, and axioms, where only one kind of variable and constant was employed. The idea was to have a *universal* language for logic, in which no a priori distinction was granted between primitive individuals and fancier constructs involving those primitive individuals. Thus for Frege, an object c and the properties P it may have were all objects to be reasoned about in the same way, i.e., with the same basic rules and notations. Frege had certain *comprehension axioms* that specifically created object-notations "P" for properties P, and stated that sentences using properties as predicates could be equivalently rephrased using properties as objects. These axioms in effect state a relationship between a name and what it names:

$$Has(c,"P") \leftrightarrow P(c)$$

or equivalently, but closer to Frege's notation:

$$c \in \{x|P(x)\} \leftrightarrow P(c)$$

In that same year Bertrand Russell showed that Frege's system was inconsistent. (Specifically, he defined the property R(x) so that $R(x) \leftrightarrow \neg Has(x,x)$, and applied this to x="R".) Russell then proposed that objects be arranged in a hierarchy with different notations and rules, thus avoiding the possibility of self-reference that led to the inconsistency in Frege's

system. The resulting "typed" system has as its first level of notations precisely that of Frege, but without the damaging axioms (the comprehension axioms) that created objects out of properties at the first level. For Russell, properties of first-level objects are to be viewed as second-level objects. Thus was born the expression "first-order logic"; it is Frege's logic except for the offending axioms. Russell's logic included this first level and also higher levels for objects coresponding to properties, properties of properties, etc. (which for Frege would have been created by comprehension axioms at this first universal level). For this reason we refer to first-order logic and the higher-order logics. In general one can choose to work with as many orders as desired. Names for objects are supplied at each level by the rules of syntactic formation.

However, some problems do arise. One is that there is a substantial burden in having to deal with a large number of different notations. This perhaps could be excused, although the apparent fact that in natural languages such as English we have no need for levels may suggest that a better approach exists. A second problem is more serious: many significant concepts cannot be expressed at all with levels, as will be seen below. The original simplicity and plausibility of Frege's approach has then continued to attract interest, and much of modern logic has been motivated by efforts to revise it to preserve its desirable features while removing inconsistency. Artificial intelligence has come to join this effort, as it became recognized that more than Russell's higher-order individuals are required in many situations. We give two examples here; others will appear as we proceed.

(i) (universal quantification) Suppose that agent A's beliefs are represented as sentences in some formal language L with levels. Then

symbols in L are indexed by their levels, eg, ti for a constant or variable or function of level i, and P/i+1/(ti) for a predicate of level i+i applied to a term of level i. Then the sentence that A has no religious beliefs, we might try to formalize as (xi)(Bel/i+1/(A1,xi) --> Not-religious-belief/i+1/(xi)). But this is not quite what the original sentence says, for we need not think any supposed religious beliefs of A to be at any particular level i. There is no way to quantify over all levels at once and stay within the framework of levels at the same time. On the other hand, we

want

to write simply (x)(Bel(A,x) --> Not-religious-belief(x)).

(ii) (existential quantification) Consider the sentence that John has a false belief. Again we might write (Exi)(Bel/i+1(John,xi) & False/i+1(xi)). But we don't know what level John's supposed false belief is at, so really we'd need to write something like (Ei)(Exi)(...), ie, for some level i John has a false belief xi. But then i is being used as a variable and so needs a level of its own, in opposition to our intention of using any level at all as a possible substitute for it. Again, we would like to write simply (Ex)(Bel(John,x) & False(x)).

To be sure, an infinite set of hierarchical sentences will do the trick in the first example: one for each value of the variable xi and index i. For instance, this is allowable in Konolige [3]. But this doesn't provide then a (single) expression that can be reasoned with. No one could ask the system the question as to whether the given assertion is the case; it would take forever to ask! Moreover, to deny the assertion would involve

a single infinitary disjunction, which is also what happens in the second example: either John has a level-1 false belief, or a level-2 false belief, or etc.

What seems to be needed is an avoidance of separate levels altogether, so that all concepts are treated at the same (first) level. For instance, McCarthy [4] has usefully introduced names for concepts (second-order objects) into a first-order system. McCarthy considers the problem of distinguishing between a phone number as a number, and a phone number as a concept (that which is dialed on a phone to reach so-and-so). This is of importance, since we don't want to say Alice knows Bill's phone number simply on the basis that she knows of the number 2345679, e.g., it may be her bank account number. Still, this may indeed be his number, and if we write Bill's-number = 2345679 then we are in trouble, for she knows the latter but not the former, or so we want to say. (Here "knows" can be taken to mean "has in mind" or "has memorized".)

McCarthy shows that this issue can be resolved by viewing Bill's-number as something different from the number itself, namely, as a second-order construct which however is embeddable in a first-order setting as a new kind of individual: a concept. That Bill's-number can be viewed as a second-order construct is seen as follows: it is a relation between Bill and a number, expressible as Phone-Number(Bill,2345679). Thus Alice may have memorized 2345679, but not the fact that this is Bill's number. The latter situation could be expressed as Knows(Alice,"Phone-Number(Bill,2345679)"), and even more usefully as

Ex [Digits(x) & Knows(Alice,concat(x,"= number(Bill)"))]

I.e., Alice knows Bill to be related to a particular string of digits, in a "Phone-Number" sort of way. This allows for the expression of such a notion as Alice knowing that Bill's number (concept thereof) is not to be found in the phone book: Knows(Alice("-(Ex)(Phone-Number(Bill,x) & Listed(x)").

Note that the Digits(x) is important, to prevent using x = "number (Bill)". Here Alice

knows

Bill's phone number in the sense of knowing

that such-and-such digits are his phone number, rather than knowing

of

his

number. She may have heard many numbers in connection with Bill: his mother's number, his work number, his house number; she knows

of

them all;

but it is his home phone number that she knows to be just that. So there is indeed an implicit sentence that she knows (believes). What Alice knows is not a number, but a fact relating Bill and a number. For this, quotation, in one form or another, is needed. See Haas [5] for another notational version, and for more detail on algorithms for reasoning in this vein. The

concat

function is a method for introducing variables into string expressions, a form of

quasi-quotation

or

quantifying in.

Concat("a","b") is interpreted as "ab", whereas concat(x,"b") remains uninterpreted until x is assigned a meaning. On the other hand, concat("x","b") is "xb", so that we have flexibility in our use of variables: opaque or transparent as desired.

McCarthy suggests that introducing function symbols for concepts (rather than quoted expressions) may be sufficient for a general treatment of concepts. However, it seems to us that this hope is unfounded. We often depend on expressions in the formation of concepts. For example, the concept of a sentence derives its usefulness from being related to particular sentences and particular words that make up sentences: "the last word you just said" is an expression which although representable as a function still refers to a particular word, not to a concept. Thus quotation seems necessarily involved at some point if we are to have a self-describing language. It appears we must describe specific expressions as carriers of (the meanings of) concepts. In any case, even the strict use of functions for general concepts will lead to paradoxical situations unless care is taken. In [5] a functional approach is taken that otherwise is along the lines suggested here.

Thus it appears reasonable to allow a certain number of levels to be "collapsed" into first-order logic, and leave the rest out (either entirely out or represented in Russell's higher types). Now, the question arises, why not collapse all levels into first-order logic, and be done with these difficulties? This however is just what causes Russell's paradox. McCarthy and others -- Elschlager [6], Weyhrauch [7], Attardi and Simi [8] -- are careful to avoid contradiction, by not using full comprehension axioms,

and indeed no need for them arises in limited cases such as these. But if we wish to address the issues raised in the introduction, and in particular the two sample sentences (i) and (ii), then we must find a way to collapse all levels into one without contradiction, i.e., we need to have a self-referential or universal language. (Indeed, Creary [9] makes some effort to carry out such a program, along somewhat hierarchical lines, and observes the need to address possible inconsistencies.)

3. Names

Because of these difficulties the approach of levels appears too restrictive for artificial intelligence in general. Montague [10] argues that modal logic is the appropriate remedy, and that this yields a consistent treatment of epistemic notions, whereas first-order logic (FOL) with the same notions, is not. But this is due to a powerful strengthening of FOL in writing variables that can be replaced by names for formulas; if his modal logic is similarly endowed, i.e., with propositional variables, then it too appears inconsistent. We will investigate this in formal detail elsewhere. See also Burge [11], for another criticism of Montague's position.

To motivate another approach, let us consider an extended example. Consider the English sentence, "John believes that Ronald Reagan is President." This we could formalize as

S1: Bel(John,"President(Ronald Reagan)")

where "Ronald Reagan" is a constant term, and our formal language has constant names for sentences (here the second argument to Bel names the sentence inside quotes). Here we have adopted the ideas of Moore and Hendrix [12], regarding beliefs as forming a set of sentences.

Now, it is essential to have also an un-naming device that would return a quoted sentence to its original (assertive) form, together with axioms stating that that is what naming and un-naming accomplish. This would make it possible, for example, for another agent, say Sally, to reason from the sentence S1 above that John has a true belief (assuming she also believes Reagan is President) or that John will answer "Ronald Reagan" when asked who is President. She could put herself in John's shoes, un-naming his beliefs, and then reason with the results, being careful to avoid using other beliefs of her own that she feels are not also ones of John's. (Haas [5] and Konolige [3] study aspects of this.)

Consider the further sentence

S2: "There is someone whom John believes to be President."

This we might try to formalize as (Ex)Bel(John,"President(x)"). Something is wrong here. The inner "x" is not recognized by the syntax of FOL; there's simply a single constant "President(x)". What we want is a way to take an arbitrary x and form from it a sentence name "President(x)". So let's do just that: let concat be a new 4-place function symbol, where concat(a,b,c,d) intuitively stands for a name of the concatenation of whatever a, b, c and d are names of. (If some argument isn't a name, concat can default to some convenient constant.) Then we can write

(Ex)(Bel(John,concat("President" , "(" , x , ")"))).

Now here an appropriate witness to John's existential belief--i.e., an object fi lling the role of the x required to exist--is not Ronald Reagan but rather "Ronald Reagan". For only the string naming Reagan concatenates with "President" etc., to give the desired name "President(Ronald Reagan)". This may appear clumsy and even counter to the sense of S2; but note that unless there is a description that John can use to refer to Reagan, then it makes little sense to say that he has the belief in question. For the squeamish, we can be a little fancier:

(Ey)(Ex)[Names(John,x,y) &

Bel(John,concat("President" , "(" , x , ")"))]

i.e., there is a person y and an object x (that John uses as a name for y) for which John believes the indicated sentence (namely, in this case, "President(Ronald Reagan)"). Here y is Reagan himself, and x is "Ronald Reagan". Note that the second extended version above actually entails the fi rst.

This provides a solution to a problem pointed out by Moore [13]. For instance, Moore mentions the following rule we may want to adopt for the predicate Knows: Knows(a,"p-->q") & Knows(a,"p") --> Knows(a,"q") where "a" stands for a person. If we want to be able to use such a rule for arbitrary p and q we must use variables in place of p and q. If we quote the variables, this could mean inventing special string matchers, as Moore warns. But using concat, it is fairly direct:

Knows(a , concat(x, "-->" ,y)) & Knows (a , x)

--> Knows(a , y).

(In fact Moore does something like this, although more complicated, along with his possible-worlds treatment; however we need not follow him that far to get what we need. The decision to represent knowledge as quoted sentences, together with variables ranging over such sentences, already holds enough for us.)

Here we have used concat with only three arguments, so we had best tell the whole story about it and about names: we really want concat to be a 2-place function symbol, and when we write concat(a,b,...,n) we are abbreviating concat(a,concat(b,(concat(...(m,n)))...). There are many ways to create names. One that is both simple and general is as follows: First we employ a form of Hollerith quotation, i.e., n:a1...an is a name for the string a1...an of the n symbols ai. These names are new "compound" constant symbols, not counted as single symbols when forming a name in which they appear. Thus a name for (x=y) is 5:(x=y), and a name for *this* is 7:5:(x=y) rather than 1:5:(x=y) even though 5:(x=y) is a single compound symbol as well as a string of seven simple symbols. Note that the colon is counted, and its name is 1::. This form of quotation is used to avoid the problem of nested quotation marks; now we have ability to name arbitrary strings made of *any* symbols in our language including those used in the quotation mechanism itself. Then we require concat(n:a1...an,m:b1...bm) = n+m:a1...bm. We note that in Haas [5] an alternative notation is used that has some simplifying advantages, although less generality. In the sequel we will however revert to quotation marks for the most part, it being understood that the Hollerith form is available to sort out ambuguities.

Now that we have motivated the need for an un-naming or un-quoting device, let us see whether it can be obtained in a form that is general, useful (practical), and consistent. Prior results by Tarski [14] and Montague [10] suggest that our goal may be unobtainable. However, logicians have continued to explore ways of capturing the intuitive sense of Frege's system without the inconsistencies, and we shall exploit and combine some of this work to achieve an apparently satisfactory treatment.

4. A new approach to truth

Names do present a difficulty however, namely that found by Russell for Frege's system. In its barest form, it amounts to Tarski's "No Truth-Definition Theorem" [14]: In general, True("A") <--> A is inconsistent, i.e., unquotation doesn't fully undo quotation. Now if we assume True("A") <--> A , then for certain cases of A -- e.g., the famous Liar sentence L : -True("L") -- we get True("L") <--> L <--> -True("L"). Indeed, from identifying True("P(c)") and Has(c,"P") we would get either of Russell's or Tarski's results from the other.

We must then decide how to eliminate such cases and yet allow benign and useful cases of self-reference. Our original goal of keeping all syntax available for inspection prevents us from simply outlawing certain expressions from the language.

Kripke [15] introduced a brave attack on this classical problem of truth-definitions. In order to avoid the consequences of Tarski's theorem to the effect that True("A")<-->A is in general inconsistent, he suggests that for some formulas A neither True("A") nor False("A") hold. This means excluded middle, in the form True("P") or False("P"), does not hold for all P in Kripke's system. While this can be regarded as a negative feature, leaving "gaps" in the truth definition, otherwise it has very intuitive behavior.

Tarski's theorem (or the Liar paradox) can be equally regarded as a variant on Russell's paradox, as we will see below. Tarski's own approach to the problem raised by his theorem was similar to Russell's: to introduce a hierarchy of truth predicates, each to apply to formulas formed at stages prior to it. This has the defect of not allowing reference to general formulas; for example True("A")-->A, although valid for each truth predicate True when applied to any formula A formed prior to the introduction of that predicate, cannot be stated in one formula for all levels of the hierarchy at once. And the statement that a particular formula B is "not true" at *any* level (-True("B")) is not representable either.

For this reason Kripke introduced his approach using truth gaps, in which there is only one truth predicate. Yet the problems persist on close inspection. Kripke himself comments on the problematic

character of gaps:

> "...Liar sentences are not true in the object language...but we are precluded from saying this in the object language by our interpretation of negation and the truth predicate...The necessity to ascend to a metalanguage may be one of the weaknesses of the present theory. The ghost of the Tarski hierarchy is still with us."

In effect, we note that for some formulas A, True("A") never appears in Kripke's construction, so that we conclude for ourselves that -True("A"), yet this latter formula is accorded no recognition in Kripke's formal apparatus. It may be that False("A") appears, in which case there is no reason for concern; but also A may be paradoxical, such as a Liar sentence, and then neither True("A") nor False("A") will appear, and the formalism doesn't record what to us is salient, namely the very fact that neither of these appeared.

In developing his method, Kripke utilized a procedure for assigning to True (and False) more and more terms until a fixed point is reached. This choice was a conscious departure from the standard (Tarskian) semantics for first-order logic. In Tarskian semantics, the "truth" of atomic formulas in some domain is determined by some external means, and then all the rest follows, including the holding of -A when A is not determined to hold. Thus Kripke's dilemma does not occur in Tarskian semantics, although other problems do seem to, as noted.

Here we want to suggest that in fact Kripke's work provides the basis for a first-order (excluded middle) treatment of truth after all, in which True is an ordinary first-order predicate symbol and "truth" in the sense of holding in models is kept to be the strictly Tarskian one. We begin by noting a simplifying characterization of Kripke's construction.

We suppose a first-order theory T to have a monadic predicate symbol True. We will interpret True(x) intuitively as meaning that the formula named by x is determined to be "true" in Kripke's sense (in some domain). Thus True(x) in particular means that x is grounded (it can be reduced to formulas not involving "True") and the result holds. Now, this is quite a complex notion, and formalizing it straightforwardly by mimicking Kripke's metalanguage construction would require in the object language a rather massive set of axioms including set theory. However, we can do it much more easily simply by formalizing the single iterative step in the construction, namely, deciding the case of True(x) by reducing it to the case of x. Kripke proceeds by saying True("A") if A has already been determined at a previous stage.

The trick follows Gilmore [16]. We simply posit

True("A") <--> (A)*

instead of the earlier-mentioned schema (without the star) where the * operator replaces each connective occurrence of the form -True("...") in A by True("-(...)"). (Here we have to be careful first to pass negation through to predicate letters, and also to rewrite conditionals A --> B as B v -A.)

Note that, in general, True("-A") is NOT equivalent to -A, and thus (as it turns out) contradictions do not arise from the famous paradoxes. Yet, we still gain the advantages of self-reference because the two expressions are equivalent in case A itself does not contain the predicate "True" -- or more generally if A is

positive

in a sense that will be defi ned later.

The paradigmatic case is the following:

True("-True('B')")<-->True("-B")

In Kripke's terms, we decide True("-True('B')") only if we already have decided -True("B"), which for Kripke can only be in the form False("B") and which we here write as True("-B") to keep the number of new predicates down. We see then how we can preserve the important property of excluded middle (A v -A) for all formulas A. With Kripke, we do not require True("A") v True("-A") for all A, but since now -A is not equivalent to True("-A"), we avoid a contradiction. True("-A") has a stronger meaning than simply -A: the former means "-A" was found to come out of Kripke's iterative defi nition, whereas the latter means simply not whatever A says. For instance, if A is True("B") then -A simply attests to the fact that B doesn't come out of Kripke's iterations (while A says that it does); but True("-A") says (-A)*, and this is True("-B"), i.e., that -B does come out of the iterations.

In most cases, the new schema True("A")<-->A* reduces immediately to the former schema True("A")<-->A, and when -True does appear in A, the rule

is still trivial to apply and intuitively sensible. For example, True("-True('1=2')") is equivalent according to the above schema to (-True("1=2"))*, which is just True("-(1=2)"). This in turn is equivalent to -(1=2). We remind the reader that a slightly different quotation mechanism is preferrable, namely Hollerith quotation as before, since then there is no scoping ambiguity, but this isn't necessary for purposes of illustration here.

Thus whenever "True" appears as a predicate letter in A*, it will not be negated, and so nothing is ever asserted to be ungrounded in A*. The schema in effect says to strip off the predicate letter True and work with the result, taking care not to allow a "gap"-like formula to appear. If indeed A is grounded, the schema will determine it to be true or false [True("A") or True("-A")] in analogy with Kripke's construction. If on the other hand the schema when applied successively never eliminates all occurrences of True, then no conclusion of this form is reached, again in line with Kripke. This does not mean that negation of "True" is disallowed in our language. On the contrary; it is simply that the algorithm for calculating (equivalents of) True("A") proceeds by first eliminating negations of "True" inside A. That this has an intuitive meaning we see from Kripke: True("A") means A is grounded, first of all, and further whatever A says. In effect Gilmore's * tells us when it makes sense to "step back" from the meaning of a formula A in order to comment on that meaning.

Now, the above schema will be shown to be consistent in the following precise sense: If T is any first-order theory, then its first-order extension T´ formed by including True as a new predicate letter, constants

("names") for all formulas A (which we usually write "A", although this may be ambiguous so that another mechanism such as Holleri

supplying new non-logical axioms, is still consistent. Moreover, this can be proved by providing a model in a step-by-step fashion paralleling Kripke's construction. But now all the fancy metalanguage is left by itself in the usual Tarskian semantics, and the object language specifies all that we need to know about True. Thus True is defined in the very language to which it applies, and it is a total predicate in the usual first-order sense. As a consequence, we now will have True("A") or -True("A") for each A, although not True("A") or True("-A"). This means simply that the paradoxical cases

are

expressible in the theory itself, as paradoxes, yet without jeopardizing consistency.

Some peculiarity must be adjusted to for the Liar sentences L, where L <--> -True("L"), since we will have neither True("L") nor True("-L"), and hence we will have -True("L") and -True(-"L"). But L is a Liar, so from -True("L") we get L! Thus we must live with L and -True("L") together. But this is fine as long as we recall that now True is taken to mean Kripke's sense, i.e., grounded and true (in the Tarskian sense).

Now, what are we to make of a sentence such as L, for which we can prove both L and also -True("L")? Is L true or isn't it? Well, it certainly

is

proven,

so in that sense it is established and in conformity with the facts of the situation; and it indeed will hold (be satisfied) in any model of the situation. It is only in the urge to call L "True" that we

must restrict ourselves, and this is because of the very special nature of L, in that it itself specifically states that it is not to be so designated. Now this is the point: if we are going to allow our language sufficient flexibility to have variables that refer to expressions of the language itself, then such sentences as L will crop up. Our approach allows this, and recognizes them as the paradoxical sentences they are, without letting this create an inconsistency. The price is simply that we stick literally with what the sentences say, and this inconvenience will be as rare as are these sentences in typical discourse situations. We will give examples of this in section 6.

5. The consistency proof

Let L be a first-order language. Consider an extension L´ of L containing the predicate symbol, True, of one argument, as well as constants naming all expressions, such as provided by Hollerith quotation. It is natural to consider as an axiom schema for a theory over L´ the following, for each term t naming a closed formula consisting of a string of symbols e1...en:

True(t) <--> e1 . . . en,

and which we ungrammatically write as True(t) <--> t. (That is, t is a form of quotation, such as n:e1...en as seen earlier, and for simplicity we also write it in place of the string it names when outside the predicate True; alternatively, at times we will write the string in place of its name, thereby leaving off quotation devices.)

However, this can lead to Russell's heterological paradox, as follows: with only a modest amount of symbol-manipulation power, such as concatenation,

one can construct a formula R(x) that intuitively says that (the formula named by) x does not apply to its own name as argument, and then R(R) will appear to assert its own denial in the form:

R(R) <--> -True(R(R)).

Definition: Call a

truth system

any first-order theory T having a designated constant <e1...en> for every formula e1...en, and a monadic predicate symbol True, such that for no term <x> are True(<x>) and True(<-x>) both theorems. (We usually will however omit the symbols < and >.)

Then by the construction of Russell's paradox we see that there does not exist a truth system (with concatenation) in which True(x) <--> x is a theorem for every closed formula x. (This can be regarded as a version of Tarski's "No Truth Definition Theorem.")

As seen above, a key construction we will borrow from Gilmore is that of x*, the "positive" form of x: Call x positive if True is not in the scope of negation in the formula resulting from passing negation signs in x through to predicate letters, following the usual valid rules for this regarding quantifiers and connectives. (It is important for these purposes that conditionals x --> y be written as y v -x.) Then let x* be the result of passing "-" through True as well, so that (-True(x))* is True(-x). Then if x is positive, we have

x <--> x*.

Theorem: If T is a consistent first-order theory then T has an extension GK(T) -- for Gilmore/Kripke -- which is a truth system with axiom schema True(x) <--> x* for all closed formulas x.

Proof (applying methods in [16] to ideas in [15]): Let M0 be a model of T, with domain D. Extend T by adding True and constant names to its language as above. M0 will still be a model of this extension if we interpret True as the null relation. We can regard M0 as determined by its true atomic formulas, i.e., those that hold there: these serve to interpret the predicate and function symbols. We will develop a model M of GK(T), where GK(T) is a truth system with axioms those of T plus the schema True(x) <--> x for positive x. We will do this by interpreting True in stages, starting (i.e., in M0) as the null relation, so that in M0 we have -True(x) for all x. As we extend the applicability of True in further models Mu, we will be automatically determining new atomic formulas which are to hold. The idea here is that -True(x) isn't necessarily permanent unless we first have decided True(-x); the latter is regarded as definite once established, while the former may change as the sense of True grows.

Now, for any ordinal u for which Mu has been defined, let Mu+1 = Mu + the set of "True(x)" for which x* is true (holds) in Mu. That is, we change the interpretation of True in Mu+1 by making True hold for some additional strings.

This requires explanation. We suppose True to be part of the underlying language, so that True(x) does not hold in M0 as noted above. Then in M1, for each atomic truth in M0, such as x=x, we get True("x=x") as an atomic truth in M1 by definition, whereas in M0 we have -True("x=x").

For limit ordinals i, with Mu defined for u<i, let Mi = UMu (u<i), where again Mu is regarded as represented by the set of its true atomic formulas.

Now the underlying language will have some cardinality k, i.e., the cardinality of its symbols, so also the number of formulas is k, and thus the sequence

$$M0 \subset M1 \subset M2 \subset ... \subset Mu \subset ...$$

must at some ordinal e become constant: Mu = Me for all u > e.

Let M = Me. This is our candidate for a model of GK(T). (Note that GK(T) has no non-first-order rules of inference, so that this shall be a model in the usual sense.)

Since our goal is to show GK(T) is a truth system, we must show (True(x) & True(-x)) is not a theorem of GK(T). This will follow if indeed M is a model of GK(T) and (True(x) & True(-x)) is false in M for all x.

First, to show M is a model for GK(T), we need only verify the axiom schema True(x) <--> x* since the other axioms already hold by virtue of M0 (and hence M) being a

model of T. So let True(x) hold in M for some x. Then True(x) already holds in some Mu, since it is atomic and M = UMu (u < e). Assume u is the least such ordinal, hence not a limit. Then x* holds in Mu-1. But positive formulas, once true, remain so in our construction (this is a simple lemma) so that also x* holds in M. Thus we have shown that True(x) --> x* holds in M.

Now we turn to the converse: x* --> True(x) in M. Let x hold in M. But M = Me = Me+1, and True(x) holds in Me+1 by construction. This gives the desired result.

We see then that True(x) <--> x* holds in M. Thus M is a model of the theory GK(T). Moreover, True(x) --> x holds in M for all x, and we will need this fact to proceed. Observe that if we had x* --> x for all x in M, then this result would follow. But in fact x* --> x for all x in M. We see this as follows: Let x be -True(y), so x* is True(-y). Then if True(-y) is true in M, then (-y)* holds in M; but inductively assuming the desired conditional for fewer instances of connectives, quantifi ers, and Trues than in x, we get (-y)* --> -y, hence also -y holds in M. Now if True(y) were true in M, then again we would have y* in M and thus y as well, contradicting -y. So we have -True(y), i.e., x, from the hypothesis x*. The more general case for arbitrary x follows similarly.

The above observation leads immediately to the

conclusion that in M, True(x) --> x for all x, since we have x* --> x as well as True(x) --> x*. Now we see that (True(x) & True(-x)) is false in M since otherwise we would have both x and -x true in M. Therefore True(x) & True(-x) cannot be a theorem of GK(T), and so GK(T) is indeed a truth system.//

This indicates that Kripke's intuitive view of truth in terms of groundedness can be treated consistently in a first-order setting, so that excluded middle is upheld, and also the truth outcomes and the lacks thereof are all expressible within the formalism.

Since a Kripke-like model serves to show consistency of the Gilmorean scheme it follows that Kripke's intuitive sense of truth respects that scheme: that "true" sentences are ones that can be suitably tied to ground formulas. We have "improved" on Kripke, not so much in the meaning of the truth predicate as in the formal status (FOL) so that excluded middle is preserved and the "gaps" of Kripke become explicitly stated as -True(x) (rather than simply failing to have True(x) or True(-x)).

6. Sample applications

A detailed treatment of applications to beliefs, concepts, and modal approaches to these questions will be pursued in a sequel. Here we consider some examples from ordinary reasoning, in which however belief and other cognitive notions are not at issue. For our first example, imagine Bill and Sue meet, and Bill begins the conversation:

"Did John talk to you about me?"

"Yes."

"Well, whatever it was, I'm sure it's a lie."

"But John told me that I can trust what you say."

This has various sorts of information being exchanged. We focus simply on the part dealing with truth, i.e., that Bill said that something, x, is false, and yet x itself is John's claim that Bill's statement is true. The fact that this is paradoxical is not what interests us at first; rather we are concerned with the representation of what is being said. In our formalism it is easily expressed:

Said(John,WJS) & Said(Bill,WBS)

& WBS = "Said(John,WJS) --> False(WJS)"

& WJS = "Said(Bill,WBS) --> True(WBS)"

where WJS and WBS are constants standing for "what John said" and "what Bill said", respectively. Since Bill does not know in advance what in fact John said, it is unreasonable to suppose John's statement to be represented as a concept at a given hierarchical level; indeed the example specifically illustrates the need for a lack of committment on this because as it turns out John's statement refers to Bill's, and so cannot support any ordinary hierarchy in which the outer statement (Bill's) must come either after or before the inner statement (John's): the two apparently must cohabitate one level.

Now to the paradoxical aspect of the example. A "naive" reading of True would in this context quickly lead to a contradiction. It is readily seen that True(WJS) implies True(WBS), and then False(WJS) follows. From this we conclude -True(WJS). Also we derive then False(WJS) and so -True(WBS) after all. All this follows the Gilmore/Kripke interpretation. But now a naive reading would in addition infer

True(-WJS) and

True(-WBS) from the above, and the latter yields False(WBS) and so -False(WJS) from the definition of WBS. But True(-WJS) yields False(WJS), contradicting -False(WJS), and the paradox would thus deluge us in inconsistency and all its usual plethora of conclusions.

On the other hand, staying with the Gilmore/Kripke approach, we find only -True(WJS) and -True(WBS), harmless enough. Even if we replace False(WJS) by -True(WJS) in the example, we only derive, in addition to -True(WJS) and -True(WBS) as before, the conclusion WBS itself (unquoted). Then the paradox -- WBS and -True(WBS) -- is revealed but still harmlessly! WBS is seen to be of the Liar type, as indeed it is, but no contradiction ensues. The odd swept under the rug; it is represented faithfully and yet under control. Any effort to prove True(WBS) -- which would lead to contradiction -- simply goes around in a circle: in terms of the consistency proof, at no ordinal in the tower of models will WBS be decided to be true: -True(WBS) holds at every level because (WBS)* never holds. Indeed, we have

WBS equivalent to -True(WBS). So WBS

and -True(WBS) both hold at all levels.

Although it is not absolutely out of the question that a sufficiently astute use of hierarchies could deal with our example, it is by no means clear what this would be. We hope to have shown that a simple approach, without use of hierarchies or the abundance of separate languages that entails, is available. Straightforward efforts to represent the example so

as to avoid self-reference (which is what hierarchies or separate languages would presumably avoid) seem fraught with difficulties. We illustrate this with further discussion of our example.

Suppose that Sue is a robot that has heard the utterances of Bill and John. If she represents them as we have done above, then of course our analysis remains unchanged. But suppose instead she regards statements by others as being in other languages than her own. Will this help her avoid a contradiction without the need for a special treatment of truth as we are urging? It seems not. For in order for her to understand what they are saying, she then must translate these utterances into her own language. Thus she may have the representations TrueB(WBS) <--> True(WBS´) and TrueJ(WJS) <--> True(WJS´), where TrueB and TrueJ are her predicate letters applying to "foreign" utterances in the languages of Bill and John, and WBS´ and WJS´ are her translations of these into her own language. Now, in order for her to reasonably be said to understand what she hears, she must also know the significance of their statements, i.e., that the truth conditions for WBS have to do with WJS, and vice versa, and she must be able to state this in her own language so as to reason about it:

WBS´ = "SaidJ(John,WJS) --> FalseJ(WJS)" = "Said(John,WJS´) --> False(WBS´)"

WJS´ = "SaidB(Bill,WBS) --> TrueB(WBS)" = "Said(Bill,WBS´) --> True(WBS´)"

where equality here is equivalence modulo Sue’s translation, and is surely somthing she must be able to utilize if she is to reason as we intend. But now Sue has reconstructed in her own language exactly the original form of the problem, and so will find the same issues as before. Unless she adopts a special treatment of truth she will derive a contradiction.

In case it may appear that we are deliberately and unnecessarily providing Sue with the ingredients for paradox, note that until she comes across that observation (of paradox) it is entirely reasonable for her to proceed as we have indicated. That is, until she has translated the utterances along the lines above, as far as she knows they may be harmless statemen said John is a nice fellow, and John may have said that Bill speaks with a lisp; these statements can be regarded as having interrelated truth conditions also, but innocuous ones. Indeed, even their original statements are harmless separately, and it would be seem overly restrictive not to allow Sue to make sense of one person saying something about another's utterance. It is only after so doing that a conflict may turn up. It appears then that the ability to reason about (the truth of) what others say, carries with it the possibility of finding statements that refer to one another and hence are (at least indirectly) self-referential.

Now, a real person Sue in the robot's shoes would probably quickly go through some reasoning along the lines we suggest, and then smile at the paradox unconcernedly, reasoning that Bill mistakenly thinks John dislikes him, and not bother further about truth assignments for the two original utterances: it simply won't matter to her, since she cares about other more significant information. Note though that it is not only a matter of focus of interest or attention. For until it is recognized that there may be a misunderstanding, either one of the statements might be true or false and it might be important to find out. Thus the determination, as well as the subsequent avoidance, of apparent co component of a formalization of such reasoning. The trick is to be able to observe the paradox and yet not be forced by it into an outright contradiction that could infect the whole reasoning process. Our approach does just that.

This is not to say that contradictions cannot be tolerated in a language

for reasoning. After all, any number of contradictory pieces of information may be presented by various sources, and initially accepted. So other means of dealing with contradiction are also important. However, in some cases, as our example shows, it is useful to be able to discount the apparent contradiction immediately. Moreover, if the conflicting information is of the self-referential variety then it is not appropriately attributed to external errors of information; it is in the language itself and should be addressed as such.

In fact, at times it is essential to tease out the inter-referentiality of statements in order to draw a perfectly legitimate and desired conclusion. In the time-honored tradition of artificial intelligence, we turn to a logical puzzle for our final example. Consider the following problem of "Od and Id" offered in the spirit of Smullyan [17], although it is simpler than those he discusses.

> Forensic psychologist Jane Crane travelled to Lower Slobbovia where she was asked to solve a case of perjury involving two suspects named Od and Id. Now in Lower Slobbovia, humans always tell the truth. It was suspected that at least one of Od and Id was not human. Crane interviewed the suspects together, and the following statements were made:
>
> Id: "We both always tell only the truth."
>
> Od: "That's not true."
>
> From this information Crane was able to determine that Id indeed

was not human.

We suggest that this can serve as a challenge for a formal reasoning system. It has an intelligible solution and involves no tricks. However, it does seem necessarily to involve statements that refer to one another to an extent that eludes a hierarchical approach. For consider how Crane might have proceeded:

> If Id is telling the truth then so is Od; but then Id's statement would not be true after all. So Id is lying (and so is not human).

The point here is that although the situation is totally improbable, it is one that people can reason about with success, deriving an unambiguous conclusion by what seem to be ordinary modes of inference in ordinary language, unencumbered with caveats or contortions. We should expect a broad and flexible automated reasoning system to be able to do the same, or at least to be able to "follow" such a train of inference, and for this it must be able to represent the reasoning.

A representation of this reasoning in our approach could be as below:

> Human(x) & Says(x,y) . --> True(y)
>
> Says(Id, WIS)
>
> Says(Od, "-True(WIS)")

where WIS is an abbreviation for "what Id says", i.e.,

> WIS = "(y)(Says(Id,y) v Says(Od,y) . --> True(y))".

It is now easy to formally carry out Crane's reasoning. From the hypothesis of True(WIS), WIS itself (unquoted) follows (given our treatment of True, for recall that True("A")<-->A*, and A*-->A). Next from this and Says(Od, "-True(WIS)") follows True("-True(WIS)"), and therefore also -True(WIS). So True(WIS) --> -True(WIS), hence -True(WIS) is proven. It now is easy to derive -Human(Id) from the first and second axioms.

We are not trying in this example to illustrate the power of the Gilmore * operator so much as the need to have a language in which mutually referring statements are expressible. The previous example also can serve, but it was paradoxical. Here we are arguing that also a straightforward and non-paradoxical conclusion can derive from such statements, and, we suggest, only when they are explicitly represented in a non-hierarchical fashion. Any attempt to untangle the mutual reference would seem to vitiate the information needed for Crane to reach her conclusion.

Note that in some sense a purely propositional account of the above is possible. For if we simply derive ourselves the needed components of the argument, namely, that if Id is human then WIS, and if WIS then -WIS, the conclusion is immediate. So the formalization

(HI --> WIS) & (WIS --> -WIS)

allows the conclusion -HI (not Human(Id)). Then of course there is no issue of self-reference. However, this misses the point, which is that it is possible to start from general information about people and what they may say, and then particularize it to cases. The propositional formalization just shown does not in fact contain the original information, and would not allow the further conlusion that, say Od is not human either if it beomes known that

Od said Id is human.

7. Conclusions

The connection between self-reference and truth is simply that to do self-reference we need names for expressions, hence quotation (of some sort) and a way to relate the names to what is named, hence unquotation (i.e., a truth predicate). It is pointless, for example, to talk about beliefs outside the context of a world in which the beliefs may be true or false. So we become concerned with the relationship between Bel(x) and True(x). Obviously there is no hard and fast connection; that is not what is claimed at all. But there is a need to be able to represent the outcome (all cases arise: believed and true, believed and false, true and not believed, false and not believed). For this a language able to express its own syntax and semantics is necessary. We hope to have provided just this.

The unfortunate tendency to view first-order variables as rigidly tied to a narrow part of the world is probably due to the impoverished examples in logic textbooks, and leads directly to the outlook that second-order constructs are needed to talk about such things as the first-order variables themselves or their class of referrents. But in fact it is quite within the spirit of first-order logic to let (first-order) variables refer to syntactic and semantic features of the self-same language.

Indeed, the famous example of "all men are mortal" already implicitly expresses the correct attitude, in the formulation (x)(Man(x) --> Mortal(x)), where x presumably ranges over the whole universe, unspecified

as that may be, except that it should include men. It needn't be restricted to organisms, physical entities, and so on; it can include societies, ideas, theories, and even all of these together. Moreover, we needn't say so in advance: we do not need to state that x above means all or any of some prepared list of "things". Rather, by means of axioms we make claims about certain values of x, such as the man-values; on the rest we remain uncommitted. Indeed the completeness theorem for first-order logic says that such a formula as above is derivable in whatever context of axioms we have, precisely if it holds in all interpretations (of the range of x and so on) in which the context at hand also holds. Thus unless stated otherwise by an axiom such as (x)Man(x), the standard first-order semantics does not make restrictive assumptions on the "intended" range of variables.

Furthermore there is no reason, for instance as in Lisp, that we cannot let a term stand for another term or even on occasion for itself. This simply is not seen very often, but is perfectly in the spirit of first-order logic and semantics. It does mean we will need a large supply of names for things, but this is no surprise; the hierarchical treatments also supply names (though in an extended language). This however involves issues of self-reference, and if given teeth with unquotation can then lead to paradox unless handled in an appropriate way such as we have indicated earlier.

Still, the consistency of the treatment of truth should not be taken as justifying the view that the beliefs of an intelligent reasoning system should be consistent. But our *theories* of its behavior should be consistent, and it may also fruitfully form its own consistent theories of

its behavior, in which case it will need a way to refer to its own syntax and semantics. Our method provides just this.

Thus although Winograd [18] is right in that the semantics of a system can depend on properties of the processes involved, Hayes [19] is right in that (first-order) logic remains adequate to the task of expressing this dependency. As we discover more about the processes, we can express them in first-order logic, using quotation when necessary. The processes need not conform at all to the proof-theoretic mechanisms of logic, but can be whatever we deem appropriate; this has no effect on our expression of them as formulas of logic.

In conclusion, compunctions about free-swinging notation copied fairly directly from natural language, with its self-reference and relation-objectification (quotation), have kept us tied to overly weak and cumbersome representations ever since Bertrand Russell's discovery of paradox in Frege's theory of sets. Given our quotation mechanism, it is hard to see what serious restrictions are placed on knowledge representation by the requirement of first-order formalism. For we are more or less always restricted to discrete notations, and our efforts in natural language to express complex concepts rely invariably on object- and relation-terms. This means we can concentrate on the

facts

we wish to express regarding

thought and action, and not be so concerned with novel mechanisms for expressing them. For what we wish to say about intelligence has straightforward expression; the difficulty is in discovering what those facts are.

Acknowledgment

This research has developed out of interests in foundational questions stimulated in me by two advisors, Martin Davis and James Allen, and further encouraged by a great many colleagues and mentors, including Chris Brown, Jerry Feldman, Alan Frisch, Andy Haas, Kurt Konolige, Henry Kyburg, Jack Minker, Nils Nilsson, and Dan Russell. It has been supported by grants from the National Science Foundation (MCS79-02971), the Alfred P. Sloan Foundation (78-4-15), and the University of Maryland (General Research Board Summer Award).

References

(1) McDermott, D. and Doyle, J. Non-monotonic logic I, Artificial Intelligence, 13, (1980) 41-72.

(2) Perlis, D. Language, computation, and reality. Ph.D. thesis, Univ. of Rochester, 1981.

(3) Konolige, K. A first-order formalization of knowledge and action for a multiagent planning system, Machine Intelligence 10, 1982, 41-72.

(4) McCarthy, J. First order theories of individual concepts and propositions, Stanford AI Machine Intelligence 9, 1979, 129-147.

(5) Haas, A. Planning mental actions. Ph.D. thesis, University of Rochester, 1982.

(6) Elschlager, B. Consistency of theories of ideas, IJCAI-79, 241-243.

(7) Weyhrauch, R. Prolegomena to a mechanized theory of formal reasoning, Artificial Intelligence, 13, (1980) 133-170.

(8) Attardi, G. and Simi, M. Consistency and completeness of OMEGA, a logic for knowledge representation. IJCAI-81, 504-510.

(9) Creary, L. Propositional attitudes: Fregean representation and simulative reasoning. IJCAI-79, 176-181.

(10) Montague, R. Syntactical treatments of modality, with corollaries on reflexion principles and finite axiomatizability, Acta Philosophica fennica, 16, (1963) 153-167.

(11) Burge, T. Epistemic paradox. J. Phil. 81, 1984, 5-29.

(12) Moore, R. and Hendrix, G. Computational models of beliefs and the semantics of belief-sentences, SRI Tech. Note 187, 1979.

(13) Moore, R. Reasoning about knowledge and action, IJCAI-77, 223-227.

(14) Tarski, A. Der Wahrheitsbegriff in den formalisierten Sprachen, Studia Philos., 1, (1936) 261-405.

(15) Kripke, S. Outline of a theory of truth, J. Phil. 72, (1975), 690-716.

(16) Gilmore, P. The consistency of partial set theory without extensionality, in T. Jech, (ed.) Axiomatic Set Theory, Amer. Math. Soc., 1974, 147-153.

(17) Smullyan, R. The Lady or the Tiger? Knopf, 1982.

(18) Winograd, T. Extended inference modes in reasoning by computer systems, Artificial Intelligence, 13, (1980) 5-26.

(19) Hayes, P. In defense of logic, IJCAI-77, 559-565.