

FINDING OUT = ACHIEVING DECIDABILITY

Manny Rayner and Sverker Janson
Swedish Institute of Computer Science
Box 1263, S-164 28 KISTA, Sweden
manny@sics.se, sverker@sics.se

ABSTRACT

We present a framework for reasoning about the concepts of "knowing what" and "finding out", in which the key concept is to identify "finding out the answer to question Q" with "achieving a situation in which Q is decidable". We give examples of how the framework can be used to formulate non-trivial problems involving the construction of plans to acquire and use information, and go on to demonstrate that these problems can often be solved by systematic application of a small set of goal-directed backward-chaining rules. In conclusion, it is suggested that systems of this kind are potentially implementable in λ -Prolog, a logic programming language based on higher-order logic.

1. INTRODUCTION

The goal of this paper is to construct a framework which will allow us to describe problems which involve acquisition and use of knowledge. The first, and most immediate, demand that we will make on this formalism is that it should be *well-defined*, that is to say possessed of a clear formal semantics; the second is that it should, at least in many of the cases where this seems intuitively reasonable, be able to support *efficient goal-directed reasoning* about these concepts. Before we go any further, however, we will pause a moment to give an example of the kind of thing we want to achieve. At this stage, we will present the problem informally: later in the paper, we will demonstrate how to describe and solve it in terms of the formalism we will be proposing.

Problem 1: finding a squash racket

I want a squash racket. I know Keith has one, and I know that he knows where it is. I can find out where it is if I can ask him. I can do this if I can find out where *he* is. I don't know where he is, but I know that his secretary does, at least during office hours. I can ring his secretary and ask her if I know her number, which I do during office hours.

Merely from the way the problem is described, it is obvious to a human that a simple solution exists. We can reason backwards from the initial goal "find out where Keith's racket is", producing a goal-tree like the one in diagram 1: what we are trying to do, then, is to justify this kind of reasoning in formal terms.

The basic notions we will use have already been described by us in an earlier paper [Rayner & Janson 88], and are closely related to Levesque's work on incomplete databases [Levesque 81, 84]. To summarize, we follow Levesque in extending first-order

logic with a modal knowledge operator K , with the intuitive significance that $K(P)$ expresses the fact that "the agent knows P ". (Actually, we will soon see that this needs to be made relative to agents and situations; we deal with that in the next section). The point is of course that this makes it possible for us to distinguish between an agent's knowing something, and its merely being true.

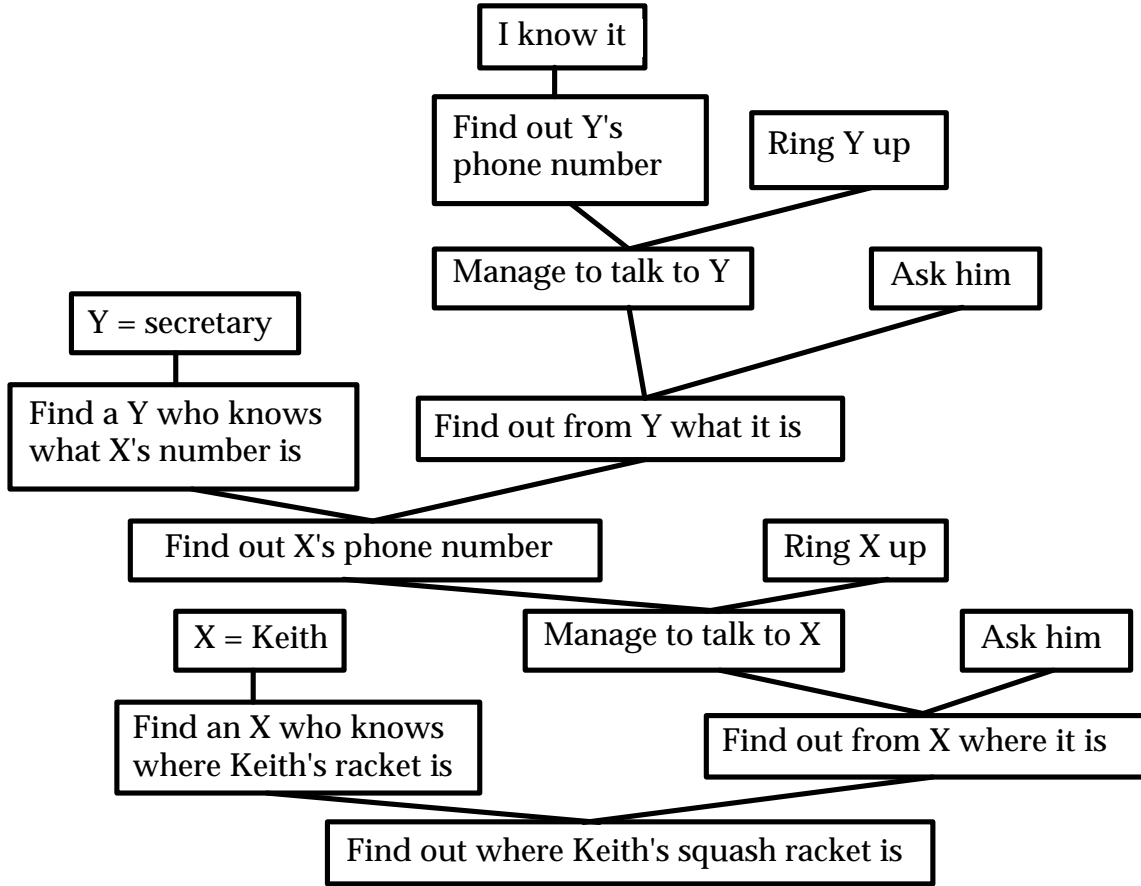


Diagram 1. Goal tree for a "finding out" problem

So far Levesque: however, we wish to extend the formalism so as to be able to talk about "knowing" as applied not just to propositions ("know the block is on the table"), but also to questions, ("know what is on the table", "know where the block is")¹. We represent questions as follows: a yes/no question is the proposition whose truth is being questioned, and a WH-question is a λ -bound form which represents the questioned property². We then define an operator KW ("knows what/whether"), which can be applied to question denotations. The intended semantics is that $KW(Q)$ is true iff Q is a decidable question. More exactly, if Q is a proposition representing a yes/no question, we have

¹The assumption we are making here (which is completely standard among theoretical linguists) is that a phrase like "know what is on the table" is composed of the verb "know" and the (embedded) question "what is on the table". We refer to the previous paper for a discussion of this idea, which may seem a little strange to readers used to the AI literature.

²Note for readers interested in NL semantics: this approach to question denotations is essentially that of [Bennett 79], and should be contrasted with the currently more popular ideas emanating from [Karttunen 77]. Once again, we refer to our previous paper for a more complete discussion of this point.

Rule 1: definition of propositional KW

$$KW(Q) \Leftrightarrow K(Q) \vee K(\neg Q)$$

and if Q is a property representing a WH-question, we have

Rule 2: definition of non-propositional KW

$$KW(Q) \Leftrightarrow \forall y.K(Q(y)) \vee K(\neg Q(y))$$

(Remember that Q is of the form $\lambda x.P(x)$.)

These are essentially the standard definitions of decidability; we refer the reader to [Rayner & Janson 88] for a detailed discussion of their applicability in the context of natural language and question answering.

The idea of the current paper is to extend the framework to cover not only the static concept of "knowing", but also the dynamic one of "finding out". We do this in the obvious way, by first relativizing "knowing" to "knowing in a situation": then we can define "finding out" by saying in effect that "Agent A can find out the answer to question Q " is equivalent with "Agent A can produce a situation S , in which Q is decidable". There are of course many ways of implementing this, depending on how we want to represent actions; in the sequel, we will content ourselves with presenting a very simple-minded solution based on the situation calculus.

Several other definitions of "knowing what" have been proposed in the AI literature, and in the interests of making our claim more precise we try to summarize briefly what we regard as the connections between the various approaches. Firstly, there is the line of investigation instigated by [McCarthy 77], which treats concepts as primitive objects in the domain; this has subsequently been examined by several other authors, most notably [Barnden 83]. In our system, the notion of a "concept" roughly corresponds to that of an intensional interpretation of λ -bound forms and modal operators; McCarthy's *denotes* predicate, which links concepts to the objects which they represent, corresponds to function application. Our opinion is that McCarthy's formalization is less elegant than to the one we propose, since he is forced to introduce pseudo-logical operators on "concepts" and ends up more or less reconstructing the λ -calculus anyway.

The second major approach is that advocated in [Konolige 84], which in effect defines the KW operator by

$$KW(Q) \Leftrightarrow \exists x.K(Q(x))$$

This definition sometimes turns out to be equivalent to ours, when the question Q is known to have exactly one answer. As we see it, our approach has the advantage of covering questions with multiple answers; for example, it seems reasonable to say "I know what is in the box" both when I know that *nothing* is in the box (no answers), and when I know that it contains (say) my wallet and passport (two answers). However, the issues involved here are sufficiently subtle that it is really more prudent to defer a serious comparison to another occasion.

The rest of the paper is laid out as follows. In section 2, we formalize the definitions outlined above and present the other mathematical apparatus we will require, and in section 3 we use it to give precise formulations and backward-chaining solutions for two typical "finding out" problems, one of them being the "squash racket" problem above. In the last section, we present our conclusions and suggest directions for further research.

2. FINDING OUT IN TERMS OF DECIDABILITY: SOME DEFINITIONS

This section summarizes the machinery we will need to formulate the examples in the next section; we start with the "knows-what" concept. We will write

$$K(\textit{Agent},\textit{Proposition},\textit{Situation})$$

for "*Agent* knows *Proposition* in *Situation*", and

$$KW(\textit{Agent},\textit{Question},\textit{Situation}),$$

for "*Agent* can decide *Question* in *Situation*". Questions will be either propositions or λ -bound forms, as described above. Following the usual formulation of the situation calculus, we write

$$\textit{res}(\textit{Action},\textit{Situation})$$

to denote "the situation that arises from an execution of *Action* in *Situation*".

We will assume that acquisition of knowledge is monotonic (once we have learned something, we can neither forget it, nor invalidate it by learning something new), and we will thus have frame axioms

$$\forall \textit{Sit},\textit{Act},\textit{Agent},\textit{Prop}.K(\textit{Agent},\textit{Prop},\textit{res}(\textit{Act},\textit{Sit})) \Leftarrow K(\textit{Agent},\textit{Prop},\textit{Sit})$$

$$\forall \textit{Sit},\textit{Act},\textit{Agent},\textit{Q}.KW(\textit{Agent},\textit{Q},\textit{res}(\textit{Act},\textit{Sit})) \Leftarrow KW(\textit{Agent},\textit{Q},\textit{Sit})$$

To give a simple example of the formalism, we present an axiomatization of the problem described in the Call for Papers. We want to say that 1) we can find out where the assembly line is by seeing it, 2) we can see it from the North side of the factory. These become

$$\forall \textit{Sit}.KW(\textit{me},\lambda X.\textit{position}(\textit{line},X),\textit{Sit}) \Leftarrow \textit{see}(\textit{me},\textit{line},\textit{Sit})$$

$$\forall \textit{Sit}.\textit{see}(\textit{line},\textit{Sit}) \Leftarrow \textit{is_at}(\textit{me},\textit{north_side},\textit{Sit})$$

If we add the assumption that we will be somewhere if we walk there,

$$\forall \textit{Sit},\textit{Place}.\textit{is_at}(\textit{me},\textit{Place},\textit{res}(\textit{go_to}(\textit{Place}),\textit{Sit})).$$

we can readily produce a backwards-chaining proof that we can in any situation find out where the line is by walking to the north side, i.e.

$$\forall \textit{Sit}.KW(\textit{me},\lambda X.\textit{position}(\textit{line},X),\textit{res}(\textit{go_to}(\textit{north_side}),\textit{Sit}))$$

Before going further, this seems as good a place as any to introduce the subject of KW expressions whose second arguments are complex formulas, i.e. formulas containing quantifiers and logical connectives; evidently, expressions of this type are going to occur in most non-trivial problems. In line with our general approach, we do not think it reasonable to hope to find a general solution which can be implemented in an efficient goal-directed way; what seems much more realistic is to look for a set of reduction rules, which will at least solve a large class of problems intuitively felt to be easy.

To be able to express these, we must first say a few words about our interpretation of the concept "backward-chaining rule", which is not entirely standard. We will use three distinct backward-chaining schemas, which we summarize briefly below:

Schema A (modus tolens)

Given a goal $P \& R$, and a rule $P \Leftarrow Q$, reduce to the goal $Q \& R$.

Schema B (\forall -introduction)

Reduce the goal $\forall x.P(x)$ to the goal $P(x^*)$, where $P(x^*)$ is derived from $P(x)$ by consistently replacing the variable x with a unique constant x^* .

Schema C (\Leftarrow -introduction)

Reduce the goal $P \Leftarrow Q$ to the goal $P \mid Q$, where $P \mid Q$ is interpreted as "Prove P , with the added assumption Q ".

We will also restrict the range of problems considered by making the important assumption that all constants and functors used are "rigid", in the sense of denoting the same individuals in each possible world. This allows us freely to substitute terms inside modal operators; however, it means that we are not permitted to use skolemization within the scope of a modal operator, as this will give rise to non-rigid designators. To use a classic example, we will thus not permit the expression

$K(\text{john}, \exists x.\text{spy}(X))$ "John believes that there is someone who is a spy."

to be rewritten using a skolem constant sk as

$K(\text{john}, \text{spy}(sk))$ "John believes that sk is a spy."

Several researchers have investigated automatic proof construction in modal logics which lack this restriction (see e.g. [Geissler & Konolige 86], [Genesereth & Nilsson 87] §§9.5–9.7). However, it is unclear to us that their methods are applicable to the type of backward-chaining proofs that we are considering here.

We now present the rules themselves, and look first at formulas of the type $KW(P\&Q)$. (In the remainder of this section, we will suppress the first and third arguments to K and KW for the sake of brevity). The following rule gives a sufficient condition for the truth of such formulas, which appears adequate in most practical cases:

Rule 3 (KW-conjunction reduction)

$KW(P \& Q) \Leftarrow KW(P) \& (KW(Q) \Leftarrow P)$

The intuitive significance of the rule is that we will know whether $P\&Q$ is true if 1) we know whether P is true, and 2) we would know whether Q was true if P was. The proof of its correctness follows directly from the definitions. There is a similar rule for disjunctions:

Rule 4 (KW-disjunction reduction)

$KW(P \vee Q) \Leftarrow KW(P) \& KW(Q)$

The trickiest case seems to be existential quantification. Intuitively, we think that people have three distinct ways of reasoning about statements of the form $KW(\exists x.P(x))$, and that these give rise to three distinct reduction rules, as follows:

Rule 5a (KW-existential to universal reduction)

$$KW(\exists x.P(x)) \Leftarrow \forall x.KW(P(x))$$

Rule 5b (KW-existential reduction to example)

$$KW(\exists x.P(x)) \Leftarrow \exists x.K(P(x))$$

Rule 5c (KW-existential reduction to lack of examples)

$$KW(\exists x.P(x)) \Leftarrow \forall x.K(\neg P(x))$$

The intuitive significance of the three rules should be clear. a) states that we can know whether there is anything which is P, if we know which things are P's; b) says that it will be enough to find one thing that is known to be P; and c) says that it also suffices to know that nothing is a P.

Finally, it is possible to expand an expression in a KW goal to an equivalent definition if the agent knows that they are equivalent, thus

Rule 6 (KW-definition expansion)

$$KW(P \& R) \Leftarrow K(P \Leftrightarrow Q) \& K(Q \& R)$$

3. TWO EXAMPLES

3.1 THE "SQUASH RACKET" PROBLEM REVISITED

We first use the formalism developed in the last section to present a rigorous formulation of the "squash racket" problem. First, in any situation Keith knows where his squash racket is:

$$KW(\text{keith}, \lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit}), \text{Sit}) \quad (1)$$

The following two axioms give a simple characterization of how one agent can acquire knowledge from another by asking a question. Firstly, the Asker will know the answer to the Question if he has successfully_asked the Asked:

$$KW(\text{Asker}, \text{Question}, \text{Sit}) \Leftarrow \text{successfully_asked}(\text{Asker}, \text{Asked}, \text{Question}, \text{Sit}) \quad (2)$$

And the asker (X for brevity) has successfully_asked the answerer (Y) the Question, if Y knows the answer to it, X asks, and they are talking together:

$$\text{successfully_asked}(X, Y, Q, \text{res}(\text{ask}(X, Y, Q), \text{Sit})) \Leftarrow \text{KW}(Y, Q, \text{Sit}) \& \text{talking_to}(X, Y, \text{Sit}) \quad (3)$$

X will be talking to Y if he has successfully performed a ring_up operation. To do this, Y needs to have a phone number, and X needs to know what it is.

$$\text{talking_to}(X,Y,\text{res}(\text{ring_up}(X,Y),\text{Sit})) \Leftarrow \quad (4)$$

$$\exists N.\text{phone_no}(Y,N,\text{Sit}) \ \&$$

$$\text{KW}(X,\lambda N.\text{phone_no}(Y,N,\text{Sit}),\text{Sit})$$

Keith has a number during office hours, and keiths_secretary knows what it is:

$$\text{KW}(\text{keiths_secretary},\lambda N.\text{phone_no}(\text{keith},N,\text{Sit}),\text{Sit}) \Leftarrow \quad (5a)$$

$$\text{in_office_hours}(\text{Sit})$$

$$\text{phone_no}(\text{keith}, \text{keiths_phone}(\text{Sit}), \text{Sit}) \Leftarrow \text{in_office_hours}(\text{Sit}) \quad (5b)$$

I know keiths_secretary's phone_no during office hours (and she has one):

$$\text{KW}(\text{me},\lambda N.\text{phone_no}(\text{keiths_secretary},N,\text{Sit}),\text{Sit}) \Leftarrow \quad (6a)$$

$$\text{in_office_hours}(\text{Sit})$$

$$\text{phone_no}(\text{keiths_secretary}, \text{secretarys_phone}(\text{Sit}), \text{Sit}) \Leftarrow \quad (6b)$$

$$\text{in_office_hours}(\text{Sit})$$

We assume that office hours are between 11:00 and 16:00, that a phone-call lasts exactly 0.1 hours, and that it takes exactly 0.01 hours to ring someone up. (Obviously this could be improved; the important point is to show that we can reason about situation-dependent information, in this case a person's phone number).

$$\text{in_office_hours}(\text{Sit}) \Leftarrow \text{time_of}(\text{Sit},N\text{-oclock}) \ \& \ 11 \leq N \leq 16 \quad (7)$$

$$\text{time_of}(11\text{-oclock},11\text{-oclock}) \quad (8)$$

$$\text{time_of}(\text{res}(\text{ring_up}(X,Y),\text{Sit}),N1\text{-oclock}) \Leftarrow \quad (9)$$

$$\text{time_of}(\text{Sit},N\text{-oclock}) \ \& \ N1 = N + 0.01$$

$$\text{time_of}(\text{res}(\text{ask}(X,Y,Q),\text{Sit}),N1\text{-oclock}) \Leftarrow \quad (10)$$

$$\text{time_of}(\text{Sit},N\text{-oclock}) \ \& \ N1 = N + 0.1$$

An axiom whose importance perhaps isn't immediately obvious: we will assume that asking a question doesn't physically affect the world. We will specifically require that it doesn't affect the positions of objects, or people's phone numbers, and that this is known to agents:

$$\text{K}(A, \text{position}(\text{Obj},\text{Loc},\text{res}(\text{ask}(X,Y,Q),\text{Sit})) \Leftrightarrow \quad (11)$$

$$\text{position}(\text{Obj},\text{Loc},\text{Sit}),$$

$$\text{S})$$

$$\text{K}(A, \text{phone_no}(\text{Person},N,\text{res}(\text{ask}(X,Y,Q),\text{Sit})) \Leftrightarrow \quad (12)$$

$$\text{phone_no}(\text{Person},N,\text{Sit}),$$

$$\text{S})$$

We want to produce a plan which lets us find out where the racket is, i.e. prove

$\exists \text{Sit.KW}(\text{me}, \lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit}), \text{Sit})$

We work as follows:

The first step is perhaps the trickiest one. We can't do the obvious thing, and simply ask where the squash racket is in Sit: if we did this, we would in the preceding situation be asking the question Q, where Q can be expressed as "Where is the racket in the situation that arises when I ask Q?". This is self-referential, and is thus disallowed. Instead, we invoke axiom (12) and find out where the racket was before we asked the question. Thus we reduce to:

$\text{KW}(\text{me}, \lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit1}), \text{Sit})$

in the process instantiating Sit to $\text{res}(\text{ask}(\text{Asker}, \text{Asked}, Q), \text{Sit1})$. We can now apply (2) to reduce to

$\text{successfully_asked}(\text{me}, \text{Asked}, \lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit1}), \text{Sit})$

and then further by (3) to

$\text{KW}(\text{Asked}, \lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit1}), \text{Sit1}) \ \& \ \text{talking_to}(\text{me}, \text{Asked}, \text{Sit1})$

instantiating Asker to me and Q to $\lambda X.\text{position}(\text{keiths_racket}, X, \text{Sit1})$. We now apply (1) to reduce the first conjunct, instantiating Asked to keith in the process, and thus leaving

$\text{talking_to}(\text{me}, \text{keith}, \text{Sit1})$

This is reduced by (4) to

$\text{phone_no}(\text{keith}, N1, \text{Sit2}) \ \& \ \text{KW}(\text{me}, \lambda N.\text{phone_no}(\text{keith}, N, \text{Sit2}), \text{Sit2})$

which instantiates Sit1 to $\text{res}(\text{ring_up}(\text{me}, \text{keith}), \text{Sit2})$

The first conjunct can immediately be reduced by an application of (5a), following which an application of (12) reduces the second one to

$\text{KW}(\text{me}, \lambda N.\text{phone_no}(\text{keith}, N, \text{Sit3}), \text{Sit2})$

and instantiates Sit2 to $\text{res}(\text{ask}(X1, Y1, Q1), \text{Sit3})$ for some so far unspecified X1, Y1, Q1, and we now use (2) again to make this

$\text{successfully_asked}(\text{me}, \text{Asked1}, \lambda N.\text{phone_no}(\text{keith}, N, \text{Sit3}), \text{Sit2})$

and then (3) to turn it into

$\text{KW}(Y1, \lambda N.\text{phone_no}(\text{keith}, N, \text{Sit3}), \text{Sit3}) \ \& \ \text{talking_to}(\text{me}, Y1, \text{Sit3})$

Using (5), the first conjunct is reduced to

$\text{in_office_hours}(\text{Sit3})$

instantiating Y1 to keiths_secretary. This has to be suspended for a while, and left as a constraint until Sit3 is completely instantiated. The rest of the proof is more or less the same as the last few steps, and we finally get

$\text{Sit3} = \text{res}(\text{ring_up}(\text{me}, \text{keiths_secretary}), 11\text{-oclock})$

and thus

Sit = res(ask(me,keith, λX .position(keiths_racket,X,Sit1)),Sit1)

Sit1 = res(ring_up(me,keith),Sit2)

Sit2 = res(ask(me,keiths_secretary, λN .phone_no(keith,N,Sit3)),Sit3)

3.2 DEALING WITH COMPLEX FORMULAS

Although it may appear involved, the "squash racket" problem is actually fairly simple, since the formulas appearing as second arguments to KW terms never contain any logical connectives. We will now present a second example, which does not have this nice property; to solve it, we will need some of the reduction rules given at the end of section 2 above.

Problem 2: Is it near me?

Given an address, I want to prove that I can find out whether or not it is within one kilometer of my house. I will assume that my city map gives the co-ordinates of any address in the rectangular box whose upper and lower sides are 8 km North of me and 4 km South of me, and whose left and right sides are 16 km West of me and 3 km East of me.

We formalize this as follows. Firstly, I know the distance between two addresses in terms of their co-ordinates. Distance will be defined by the usual Pythagorean formula:

$$\begin{aligned} K(\text{me}, \text{distance}(A1,A2,D) \Leftrightarrow & \hspace{15em} (13) \\ & \exists C1,C2.\text{co_ords}(A1,C1) \ \& \ \text{co_ords}(A2,C2) \ \& \\ & \text{pythagoras}(C1,C2,D), \\ \text{Sit}) & \end{aligned}$$

I know the definition of the Pythagorean formula too:

$$\begin{aligned} K(\text{me}, \text{pythagoras}(C1,C2,D) \Leftrightarrow & \hspace{15em} (14) \\ & \exists X1,X2,Y1,Y2.C1 = \langle X1,Y1 \rangle \ \& \ C2 = \langle X2,Y2 \rangle \ \& \\ & D = \sqrt{(X2-X1)^2 + (Y2-Y1)^2}, \\ \text{Sit}) & \end{aligned}$$

I know the co-ordinates of my own address, which we for convenience take to be the origin.

$$K(\text{me},(\text{co-ords}(\text{me},C) \Leftrightarrow C = \langle 0,0 \rangle),\text{Sit}) \hspace{15em} (15)$$

I can always decide whether any arithmetical formula (an equality or inequality) is correct:

$$KW(\text{me}, X < Y, \text{Sit}) \hspace{15em} (16a)$$

$$KW(\text{me}, X = Y+Z, \text{Sit}) \hspace{15em} (16b)$$

(and so on...)

I know what the co-ordinates of an address are, if I look it up on a map and they are within the area covered by that map:

$$KW(me,co_ords(A,C),res(look_up(A,Map),Sit)) \Leftarrow on_map(C,Map) \quad (17)$$

The limits of the city map are as described above:

$$on_map(\langle X,Y \rangle,stockholm_map) \Leftarrow -16 \leq X \leq 16 \ \& \ -4 \leq Y \leq 4 \quad (18)$$

Given an address A^* , I want to find whether it is within 1 km of my house, i.e. prove:

$$\exists Sit.KW(me,\exists D.distance(me,A^*,D) \ \& \ 0 \leq D \leq 1,Sit)$$

The first step is to remove the internal existential quantifier from the formula. We convert it into an external universal quantifier using rule 5a; this gives the (technically more general) goal

$$\forall D.KW(me,distance(me,A^*,D) \ \& \ 0 \leq D \leq 1,Sit)$$

We then remove the universal quantifier, using Schema C to replace the quantified variable D with a unique constant D^* and get

$$KW(me,distance(me,A^*,D^*) \ \& \ 0 \leq D^* \leq 1,Sit)$$

Using (16) together with Rule 3, we can remove the second conjunct and treat it as an assumption. This gives us the new (conditional) goal

$$KW(me,distance(me,A^*,D^*),Sit) \mid 0 \leq D^* \leq 1$$

We now use Rule 6 with (13) to expand the formula, getting

$$KW(me,\exists C1,C2.co_ords(me,C1) \ \& \ co_ords(A^*,C2) \ \& \ pythagoras(C1,C2,D^*),Sit) \mid 0 \leq D^* \leq 1$$

from which we can remove the existential quantifier to get

$$KW(me,co_ords(me,C1^*) \ \& \ co_ords(A^*,C2^*) \ \& \ pythagoras(C1^*,C2^*,D^*),Sit) \mid 0 \leq D^* \leq 1$$

Rule 6 again, this time with (15), now allows us to expand the first conjunct and replace $C1^*$ with $\langle 0,0 \rangle$, giving

$$KW(me,co_ords(A^*,C2^*) \ \& \ pythagoras(\langle 0,0 \rangle,C2^*,D^*),Sit) \mid 0 \leq D^* \leq 1$$

We use (14) to expand the second conjunct, remove the existential quantifier, and simplify to get

$$KW(me,co_ords(A^*,\langle X2^*,Y2^* \rangle) \ \& \ D^* = \sqrt{(X2^*)^2 + (Y2^*)^2}, Sit) \mid 0 \leq D^* \leq 1$$

Now by (16) and Rule 3, we can move the second conjunct over to treat it as an assumption:

$$KW(me,co_ords(A^*,\langle X2^*,Y2^* \rangle),Sit) \mid 0 \leq D^* \leq 1, D^* = \sqrt{(X2^*)^2 + (Y2^*)^2}$$

We finally apply (17). We will have for some $Map, Sit1$ that

$$KW(me,co_ords(A^*,\langle X2^*,Y2^* \rangle),res(look_up(A^*,Map),Sit1))$$

if we can prove

$$\text{on_map}(\langle X2^*, Y2^* \rangle, \text{Map}) \mid 0 \lesssim D^* \lesssim 1, D^* = \sqrt{(X2^*)^2 + (Y2^*)^2}$$

Using (18), we can instantiate Map to stockholm_map and reduce the goal to

$$-16 \lesssim X2^* \lesssim 3 \ \& \ -4 \lesssim Y2^* \lesssim 8 \mid 0 \lesssim D^* \lesssim 1, D^* = \sqrt{(X2^*)^2 + (Y2^*)^2}$$

This is evidently true, as can be seen from diagram 2.

As far as we know, at least some of the existing mathematical manipulation systems should be capable of performing the last few steps to complete the derivation. However, this is not the point: the interesting thing, in our opinion, is that we have succeeded in reducing the problem to this form using only backwards-chaining rules and natural definitions, in the process deriving the desired conclusion that we can obtain the required information by looking up address A* on the Stockholm map. Moreover (just as in the first example) the formal argument closely follows the intuitive one, although it is not a simple matter to draw a goal-tree.

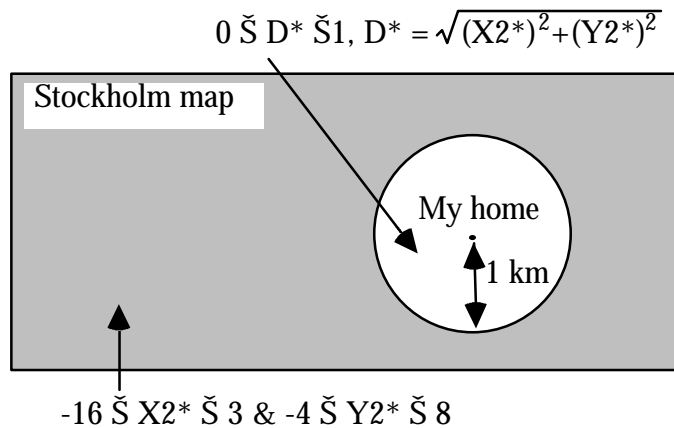


Diagram 2. The right-hand inequalities imply the left-hand ones

4. SUMMARY AND FURTHER DIRECTIONS

To sum up: we claim that we have presented a fairly robust framework for reasoning about the concepts of "knowing what" and "finding out what", based on the well-understood mathematical concept of decidability. In general, construction of proofs requires a quantified modal-logic theorem-prover; this is just as it should be, since there is no reason we can think of to suppose that *general* "finding-out" problems need be easy. It would be interesting to investigate in this context the use of a system like those described in [Geissler & Konolige 86] or [Wallen 87], and this can also form the basis of a possible line of investigation. However, our main claim is that many of the problems which intuitively seem simple can be solved efficiently by backward chaining, as one would intuitively expect. This seems to give a system which is readily applicable to practical problems in planning and natural-language understanding.

We have intentionally been somewhat vague about how a backwards-chaining system of this type could be implemented in practice, though it has hopefully been clear that we have had some kind of Prolog-like system in mind. In fact, it seems that λ -Prolog [Miller & Nadathur 86] gives us nearly the functionality we need: according to Miller (personal communication), the next version should be such that most proofs of the kind we describe could be generated automatically. We will hopefully be in a position to begin experimenting with this towards the end of 1989; in the meantime, we will continue to investigate the theoretical aspects of the problem by performing paper deductions like

those above. One point which is still very unclear to us, and which certainly needs more study, is how to formalize "uniqueness" information: that is, how to exploit facts like "a person has a unique name" or "a name and a telephone number together determine a person uniquely". However, as we have shown above, there are a great many interesting problems which can be solved without using knowledge of this kind.

In conclusion, we think that there are good reasons to support our claim that this is a useful way of characterizing the concept of "finding out".

REFERENCES

- [Barnden 83] J.A. Barnden, Intensions as Such: An Outline, *Proc. 8th IJCAI* (pp. 280–286), Karlsruhe, 1983
- [Bennett 79] M. Bennett, *Questions in Montague Grammar*, University of Indiana Linguistics Club, 1979.
- [Geissler & Konolige 86] C. Geissler & K. Konolige, A Resolution Method for Quantified Modal Logic, in J. Halpern (ed.) *Theoretical Aspects of Reasoning about Knowledge*, Morgan Kaufmann, 1986
- [Genesereth & Nilsson 87] M. Genesereth & N. Nilsson *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, 1987
- [Konolige 84] K. Konolige, *A Deduction Model of Belief and Its Logics*, Ph.D. thesis, Stanford University, 1984
- [Karttunen 77] L. Karttunen, Syntax and Semantics of Questions, *Linguistics and Philosophy* **1**, pp. 3–44, 1977.
- [Levesque 81] H. Levesque, *A Formal Treatment of Incomplete Knowledge Bases*, Ph.D. thesis, Dept. of Computer Science, University of Toronto, 1981
- [Levesque 84] H. Levesque, The Logic of Incomplete Knowledge Bases, in M.L. Brodie, J. Mylopoulos and J.W. Schmidt (eds.) *On Conceptual Modelling*, Springer, 1984
- [McCarthy 77] J. McCarthy, Epistemological Problems of Artificial Intelligence, *Proc. 4th IJCAI*, (pp. 1038-1044), Cambridge, MA, 1977
- [Miller & Nadathur 86] D. Miller & G. Nadathur, Higher-Order Logic Programming, *Proc. 3rd Intl. Conf. on Logic Programming*, (p. 448-462), Springer, 1986
- [Rayner & Janson 88] M. Rayner & S. Janson, Epistemic Reasoning, Logic Programming, and the Interpretation of Questions, in V. Dahl & P. Saint-Dizier (eds.) *Natural Language Understanding and Logic Programming II*, North-Holland, 1988.
- [Wallen 87] L. Wallen, *Automated Theorem-Proving in Non-Classical Logics: Efficient Matrix Proof Methods for Modal and Classical Logics*, Ph.D. thesis, University of Edinburgh, 1987