PROGRAMS WITH COMMON SENSE

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1959

1 Introduction

Interesting work is being done in programming computers to solve problems which require a high degree of intelligence in humans. However, certain elementary verbal reasoning processes so simple that they can be carried out by any non-feeble minded human have yet to be simulated by machine programs.

This paper will discuss programs to manipulate in a suitable formal language (most likely a part of the predicate calculus) common instrumental statements. The basic program will draw immediate conclusions from a list of premises. These conclusions will be either declarative or imperative sentences. When an imperative sentence is deduced the program takes a corresponding action. These actions may include printing sentences, moving sentences on lists, and reinitiating the basic deduction process on these lists.

Facilities will be provided for communication with humans in the system via manual intervention and display devices connected to the computer.

The *advice taker* is a proposed program for solving problems by manipulating sentences in formal languages. The main difference between it and other programs or proposed programs for manipulating formal languages (the *Logic Theory Machine* of Newell, Simon and Shaw and the Geometry Program of Gelernter) is that in the previous programs the formal system was the subject matter but the heuristics were all embodied in the program. In this program the procedures will be described as much as possible in the language itself and, in particular, the heuristics are all so described.

The main advantages we expect the *advice taker* to have is that its behavior will be improvable merely by making statements to it, telling it about its symbolic environment and what is wanted from it. To make these statements will require little if any knowledge of the program or the previous knowledge of the *advice taker*. One will be able to assume that the *advice taker* will have available to it a fairly wide class of immediate logical consequences of anything it is told and its previous knowledge. This property is expected to have much in common with what makes us describe certain humans as having *common sense*. We shall therefore say that *a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows.*

The design of this system will be a joint project with Marvin Minsky, but Minsky is not to be held responsible for the views expressed here.[1]

Before describing the *advice taker* in any detail, I would like to describe more fully our motivation for proceeding in this direction. Our ultimate objective is to make programs that learn from their experience as effectively as humans do. It may not be realized how far we are presently from this objective. It is not hard to make machines learn from experience to make simple changes in their behavior of a kind which has been anticipated by the programmer. For example, Samuel has included in his checker program facilities for improving the weights the machine assigns to various factors in evaluating positions. He has also included a scheme whereby the machine remembers games it has played previously and deviates from its previous play when it finds a position which it previously lost. Suppose, however, that we wanted an improvement in behavior corresponding, say, to the discovery by the machine of the principle of the opposition in checkers. No present or presently proposed schemes are capable of discovering phenomena as abstract as this.

If one wants a machine to be able to discover an abstraction, it seems most likely that the machine must be able to represent this abstraction in

[1]1996: This was wishful thinking. Minsky's approach to AI was quite different.

some relatively simple way.

There is one known way of making a machine capable of learning arbitrary behavior; thus to anticipate every kind of behavior. This is to make it possible for the machine to simulate arbitrary behaviors and try them out. These behaviors may be represented either by nerve nets (Minsky 1956), by Turing machines (McCarthy 1956), or by calculator programs (Friedberg 1958). The difficulty is two-fold. First, in any of these representations the density of interesting behaviors is incredibly low. Second, and even more important, small interesting changes in behavior expressed at a high level of abstraction do not have simple representations. It is as though the human genetic structure were represented by a set of blue-prints. Then a mutation would usually result in a wart or a failure of parts to meet, or even an ungrammatical blue-print which could not be translated into an animal at all. It is very difficult to see how the genetic representation scheme manages to be general enough to represent the great variety of animals observed and yet be such that so many interesting changes in the organism are represented by small genetic changes. The problem of how such a representation controls the development of a fertilized egg into a mature animal is even more difficult.

In our opinion, a system which is to evolve intelligence of human order should have at least the following features:

1. All behaviors must be representable in the system. Therefore, the system should either be able to construct arbitrary automata or to program in some general purpose programming language.

2. Interesting changes in behavior must be expressible in a simple way.

3. All aspects of behavior except the most routine must be improvable. In particular, the improving mechanism should be improvable.

4. The machine must have or evolve concepts of partial success because on difficult problems decisive successes or failures come too infrequently.

5. The system must be able to create subroutines which can be included in procedures as units. The learning of subroutines is complicated by the fact that the effect of a subroutine is not usually good or bad in itself. Therefore, the mechanism that selects subroutines should have concepts of interesting or powerful subroutine whose application may be good under suitable conditions.

Of the 5 points mentioned above, our work concentrates mainly on the second. We base ourselves on the idea that: *In order for a program to be capable of learning something it must first be capable of being told it.* In fact, in the early versions we shall concentrate entirely on this point and attempt to achieve a system which can be told to make a specific improvement in its behavior with no more knowledge of its internal structure or previous knowledge than is required in order to instruct a human. Once this is achieved, we may be able to tell the *advice taker* how to learn from experience.

The main distinction between the way one programs a computer and modifies the program and the way one instructs a human or will instruct the *advice taker* is this: A machine is instructed mainly in the form of a sequence of imperative sentences; while a human is instructed mainly in declarative sentences describing the situation in which action is required together with a few imperatives that say what is wanted. We shall list the advantages of of the two methods of instruction.

Advantages of Imperative Sentences

1. A procedure described in imperatives is already laid out and is carried out faster.

2. One starts with a machine in a basic state and does not assume previous knowledge on the part of the machine.

Advantages of Declarative Sentences

1. Advantage can be taken of previous knowledge.

2. Declarative sentences have logical consequences and it can be arranged that the machine will have available sufficiently simple logical consequences of what it is told and what it previously knew.

3. The meaning of declaratives is much less dependent on their order than is the case with imperatives. This makes it easier to have afterthoughts.

4. The effect of a declarative is less dependent on the previous state of the system so that less knowledge of this state is required on the part of the instructor.

The only way we know of expressing abstractions (such as the previous example of the opposition in checkers) is in language. That is why we have decided to program a system which reasons verbally.

2 The Construction of the Advice Taker

The *advice taker* system has the following main features:

1. There is a method of representing expressions in the computer. These expressions are defined recursively as follows: A class of entities called terms is defined and a term is an expression. A sequence of expressions is an expression. These expressions are represented in the machine by list structures (Newell and Simon 1957).

2. Certain of these expressions may be regarded as declarative sentences in a certain logical system which will be analogous to a universal Post canonical system. The particular system chosen will depend on programming considerations but will probably have a single rule of inference which will combine substitution for variables with modus ponens. The purpose of the combination is to avoid choking the machine with special cases of general propositions already deduced.

3. There is an *immediate deduction routine* which when given a set of premises will deduce a set of immediate conclusions. Initially, the immediate deduction routine will simply write down all one-step consequences of the premises. Later, this may be elaborated so that the routine will produce some other conclusions which may be of interest. However, this routine will not use semantic heuristics; i.e., heuristics which depend on the subject matter under discussion.

The intelligence, if any, of the advice taker will not be embodied in the immediate deduction routine. This intelligence will be embodied in the procedures which choose the lists of premises to which the immediate deduction routine is to be applied. Of course, the program should never attempt to apply the immediate deduction routine simultaneously to the list of everything it knows. This would make the deduction routine take too long.

4. Not all expressions are interpreted by the system as declarative sentences.Some are the names of entities of various kinds. Certain formulas represent *objects*. For our purposes, an entity is an object if we have something to say about it other than the things which may be deduced from the

form of its name. For example, to most people, the number 3812 is not an object: they have nothing to say about it except what can be deduced from its structure. On the other hand, to most Americans the number 1776 is an object because they have filed somewhere the fact that it represents the year when the American Revolution started. In the *advice taker* each object has a *property list* in which are listed the specific things we have to say about it. Some things which can be deduced from the name of the object may be included in the property list anyhow if the deduction was actually carried out and was difficult enough so that the system does not want to carry it out again.

5. Entities other than declarative sentences which can be represented by formulas in the system are individuals, functions, and programs.

6. The program is intended to operate cyclically as follows. The immediate deduction routine is applied to a list of premises and a list of individuals. Some of the conclusions have the form of imperative sentences. These are obeyed. Included in the set of imperatives which may be obeyed is the routine which deduces and obeys.

We shall illustrate the way the *advice taker* is supposed to act by means of an example. Assume that I am seated at my desk at home and I wish to go to the airport. My car is at my home also. The solution of the problem is to walk to the car and drive the car to the airport. First, we shall give a formal statement of the premises the *advice taker* uses to draw the conclusions. Then we shall discuss the heuristics which cause the *advice taker* to assemble these premises from the totality of facts it has available. The premises come in groups, and we shall explain the interpretation of each group.

1. First, we have a predicate "at". "$at(x, y)$" is a formalization of "$xisaty$". Under this heading we have the premises

$$at(I, desk) \tag{1}$$

$$at(desk, home) \tag{2}$$

$$at(car, home) \tag{3}$$

$$at(home, county) \tag{4}$$

$$at(airport, county) \tag{5}$$

We shall need the fact that the relation "*at*" is transitive which might be written directly as

$$at(x,y), at(y,z) \rightarrow at(x,z) \tag{6}$$

or alternatively we might instead use the more abstract premises

$$transitive(at) \tag{7}$$

$$transitive(u) \rightarrow (u(x,y), u(y,z) \rightarrow u(x,z)) \tag{8}$$

from which (6) can be deduced.

2. There are two rules concerning the feasibility of walking and driving.

$$walkable(x), at(y,x), at(z,x), at(I,y) \rightarrow can(go(y,z,walking)) \tag{9}$$

$$drivable(x), at(y,x), at(z,x), at(car,y), at(I,car) \rightarrow can(go(y,z,driving)) \tag{10}$$

There are also two specific facts

$$walkable(home) \tag{11}$$

$$drivable(county) \tag{12}$$

3. Next we have a rule concerned with the properties of going.

$$did(go(x,y,z)) \rightarrow at(I,y) \tag{13}$$

4. The problem itself is posed by the premise:

$$want(at(I,airport)) \tag{14}$$

5. The above are all the premises concerned with the particular problem. The last group of premises are common to almost all problems of this sort. They are:

$$(x \rightarrow can(y)), (did(y) \rightarrow z) \rightarrow canachult(x,y,z) \tag{15}$$

The predicate "$canachult(x,y,z)$" means that in a situation to which x applies, the action y can be performed and ultimately brings about a situation to which z applies. A sort of transitivity is described by

$$canachult(x,y,z), canachult(z,u,v) \rightarrow canachult(x, prog(y,u), v). \tag{16}$$

Here $prog(u, v)$ is the program of first carrying out u and then v. (Some kind of identification of a single action u with the one step program $prog(u)$ is obviously required, but the details of how this will fit into the formalism have not yet been worked out).

The final premise is the one which causes action to be taken.

$$x, canachult(x, prog(y, z), w), want(w) \to do(y) \tag{17}$$

The argument the *advice taker* must produce in order to solve the problem deduces the following propositions in more or less the following order:

1. $at(I, desk) \to can(go(desk, car, walking))$
2. $at(I, car) \to can(go(home, airport, driving))$
3. $did(go(desk, car, walking)) \to at(I, car)$
4. $did(go(home, airport, driving)) \to at(I, airport)$
5. $canachult(at(I, desk), go(desk, car, walking), at(I, car))$
6. $canachult(at(I, car), go(home, airport, driving), at(I, airport))$
7. $canachult(at(I, desk), prog(go(desk, car, walking),$
   $go(home, airport, driving)) \to at(I, airport))$
8. $do(go(desk, car, walking))$

The deduction of the last proposition initiates action.

The above proposed reasoning raises two major questions of heuristic. The first is that of how the 17 premises are collected, and the second is that of how the deduction proceeds once they are found. We cannot give complete answers to either question in the present paper; they are obviously not completely separate since some of the deductions might be made before some of the premises are collected. Let us first consider the question of where the 17 premises came from.

First of all, we assert that except for the 14th premise $want(at(I, airport))$, which sets the goal, and the 1st premise $at(I, desk)$, which we shall get from a routine which answers the question "$whereamI''$, *all the premises can reasonably be expected to be specifically present in the memory* of a machine which has competence of human order in finding its way around. That is, none of them are so specific to the problem at hand that assuming their presence in memory constitutes an anticipation of this particular problem or of a class of problems narrower than those which any human can expect to have previously solved. We must impose this requirement if we are to be able to say that the *advice taker* exhibits *common sense*.

On the other hand, while we may reasonably assume that the premises are in memory, we still have to describe how they are assembled into a list by themselves to which the deduction routine may be applied. Tentatively, we expect the *advice taker* to proceed as follows: initially, the sentence "$want(at(I, airport))$" is on a certain list L, called the main list, all by itself. The program begins with an observation routine which looks at the main list and puts certain statements about the contents of this list on a list called "observations of the main list". We shall not specify at present what all the possible outputs of this observation routine are but merely say that in this case it will observe that "the only statement on L has the form $'want(u(x))'$." (We write this out in English because we have not yet settled on a formalism for representing statements of this kind). The "deduce and obey" routine is then applied to the combination of the "observations of the main list" list, and a list called the "standing orders list". This list is rather small and is never changed, or at least is only changed in major changes of the advice taker. The contents of the "standing orders" list has not been worked out, but what must be deduced is the extraction of certain statements from property lists. Namely, the program first looks at "$want(at(I, airport))$" and attempts to copy the statements on its property list. Let us assume that it fails in this attempt because "$want(at(I, airport))$" does not have the status of an object and hence has no property list. (One might expect that if the problem of going to the airport has arisen before, "$want(at(I, airport))$" would be an object, but this might depend on whether there were routines for generalizing previous experience that would allow something of general use to be filed under that heading). Next in order of increasing generality the machine would see if anything were filed under "$want(at(I, x))$" which would deal with the general problem of getting somewhere. One would expect that premises (6), (or (7) and (8)), (9), (10), (13), would be so filed. There would also be the formula

$$want(at(I, x)) \rightarrow do(observe(whereamI))$$

which would give us premise (1). There would also be a reference to the next higher level of abstraction in the goal statement which would cause a look at the property list of "$want(x)$". This would give us (15), (16), and (17).

We shall not try to follow the solution further except to remark that on the property list of "$want(at(I, x))$" there would be a rule that starts with the premises "$at(I, y)$" and "$want(I, x)$" and has as conclusion a search for the property list of "$go(y, x, z)$". This would presumably fail, and then there would have to be heuristics that would initiate a search for a y such that "$at(I, y)$" and "$at(airport, y)$". This would be done by looking on the property lists of the origin and the destination and working up. Then premise (10) would be found which has as one of its premises $at(I, car)$. A repetition of the above would find premise (9), which would complete the set of premises since the other "*at*" premises would have been found as by-products of previous searches.

We hope that the presence of the heuristic rules mentioned on the property lists where we have put them will seem plausible to the reader. It should be noticed that on the higher level of abstraction many of the statements are of the stimulus-response form. One might conjecture that division in man between conscious and unconscious thought occurs at the boundary between stimulus-response heuristics which do not have to be reasoned about but only obeyed, and the others which have to serve as premises in deductions.

We hope to formalize the heuristics in another paper before we start programming the system.

3 References

Friedberg, R. (1958). A Learning Machine, Part I *IBM Journal of Research and Development 2*, No. 1.

McCarthy, John (1956). The Inversion of Functions Defined by Turing Machines, in *Automata Studies, Annals of Mathematical Study No. 34,* Princeton, pp. 177–181.

Minsky, M.L. (1956). Heuristic Aspects of the Artificial Intellegence Problem. *Lincoln Laboratory Report*,pp.34–55.

Newell, A., Shaw, J. C. and Simon, H.A.(1957). Empirical Explorations of the Logic Theory Machine. A case Study in Heuristic. *Proceedings of the Western Joint Computer Conference*, published by the Institute of Radio Engineers, New York, 1957, pp. 218–230.

4 DISCUSSION OF THE PAPER

PROF. Y. BAR-HILLEL: Dr. McCarthy's paper belongs in the Journal of Half-Baked Ideas, the creation of which was recently proposed by Dr. I. J. Good. Dr. McCarthy will probably be the first to admit this. Before he goes on to bake his ideas fully, it might be well to give him some advice and raise some objections. He himself mentions some possible objections, but I do not think that he treats them with the full consideration they deserve; there are others he does not mention.

For lack of time, I shall not go into the first part of his paper, although I think that it contains a lot of highly unclear philosophical, or pseudo-philosophical assumptions. I shall rather spend my time in commenting on the example he works out in his paper at some length. Before I start, let me voice my protest against the general assumption of Dr. McCarthy — slightly caricatured — that a machine, if only its program is specified with a sufficient degree of carelessness, will be able to carry out satisfactory even rather difficult tasks.

Consider the assumption that the relation he designates by *at* is transitive (page 7). However, since he takes both "$at(I, desk)$" and "$at(desk, home)$" as premises, I presume – though this is never made quite clear – that *at* means something like being-a-physical-part or in-the-immediate-spatial-neighborhood-of. But then the relation is clearly not transitive. If A is in the immediate spatial neighborhood of B and B in the immediate spatial neighborhood of C then A need not be in the immediate spatial neighborhood of C. Otherwise, everything would turn out to be in the immediate spatial neighborhood of everything, which is surely not Dr. McCarthy's intention. Of course, starting from false premises, one can still arrive at right conclusions. We do such things quite often, and a machine could do it. But it would probably be bad advice to allow a machine to do such things consistently.

Many of the other 23 steps in Dr. McCarthy's argument are equally or more questionable, but I don't think we should spend our time showing this in detail. My major question is the following: On page 9 McCarthy states that a machine which has a competence of human order in finding its way around will have almost all the premises of the argument stored in its memory. I am at a complete loss to understand the point of this remark. If Dr. McCarthy wants to say no more than that a machine, in order to behave like a human being, must have the knowledge of a human being, then this is surely not a very important remark to make. But if not, what was the intention of this remark?

The decisive question how a machine, even assuming that it will have somehow countless millions of facts stored in its memory, will be able to pick out those facts which will serve as premises for its deduction is promised to receive its treatment in another paper, which is quite right for a half-baked idea.

It sounds rather incredible that the machine could have arrived at its conclusion — which, in plain English, is "Walk from your desk to your car!" — by sound deduction. This conclusion surely could not possibly follow from the premise in any serious sense. Might it not be occasionally cheaper to call a taxi and have it take you over to the airport? Couldn't you decide to cancel your flight or to do a hundred other things? I don't think it would be wise to develop a programming language so powerful as to make a machine arrive at the conclusion Dr. McCarthy apparently intends it to make.

Let me also point out that in the example the time factor has never been mentioned, probably for the sake of simplicity. But clearly this factor is here so important that it could not possibly be disregarded without distorting the whole argument. Does not the solution depend, among thousands of other things, also upon the time of my being at my desk, the time at which I have to be at the airport, the distance from the airport, the speed of my car, etc.

To make the argument deductively sound, its complexity will have to be increased by many orders of magnitude. So long as this is not realized, any discussions of machines able to perform the deductive — and inductive! — operations necessary for treating problems of the kind brought forward by Dr. McCarthy is totally pointless. The gap between Dr. McCarthy's general programme (with which I have little quarrel, after discounting its "philosophical" features) and its execution even in such a simple case as the one discussed seems to me so enormous that much more has to be done to persuade me that even the first step in bridging this gap has already been taken.

DR. O. G. SELFRIDGE: I have a question which I think applies to this. It seems to me in much of that work, the old absolutist Prof. Bar-Hillel has really put his finger on something; he is really worried about the deduction actually made. He seemed really to worry that that system is not consistent, and he made a remark that conclusions should not be drawn from false premises. In my experience those are the only conclusions that have ever been drawn. I have never yet heard of someone drawing correct conclusions from correct premises. I mean this seriously. This, I think is Dr. Minsky's point this morning. What this leads to is that the notion of deductive logic being something sitting there sacred which you can borrow for particularly sacred uses and producing inviolable results is a lot of nonsense. Deductive logic is inferrred as much as anything else. Most women have never inferred it, but they get along pretty well, marrying happy husbands, raising happy children, without ever using deductive logic at all. My feeling is that my criticism of Dr. McCarthy is the other way. He assumes deductive logic, whereas in fact that is something to be concocted.

This is another important point which I think Prof. Bar-Hillel ignores in this, the criticism of the programme should not be as to whether it is logically consistent, but only will he be able to wave it around saying "this in fact works the way I want it". Dr. McCarthy would be the first to admit that his proramme is not now working, so it has to be changed. Then can you make the changes in the programme to make it work? That has nothing to do with logic. Can he amend it in such a way that it includes the logic as well as the little details of the programme? Can he manage in such a way that it works the way he does? He said at the begining of his talk that when he makes an arbitrary change in the programme it will not work usually, evidence, to me at least, that small changes in his programme will not obviously make the programme work and might even improve it. His next point is whether he can make small changes that in fact make it work. That is what we do not know yet.

PROF. Y. BAR-HILLEL: May I ask whether you could thrash this out with Dr. McCarthy? It was my impression that Dr. McCarthy's advice taker was meant to be able, among other things, to arrive at a certain conclusion from appropriate premises by faultless deductive reasoning. If this is still his programme, then I think your defence is totally beside the point.

DR. O. G. SELFRIDGE: I am not defending his programme, I am only defending him.

DR. J. McCARTHY: Are you using the word ‘programme’ in the technical sense of a bunch of cards or in the sense of a project that you get money for?

PROF. Y. BAR-HILLEL: When I uttered my doubts that a machine working under the programme outlined by Dr. McCarthy would be able to do what he expects it to do, I was using ‘programme’ in the technical sense.

DR. O. G. SELFRIDGE: In that case your criticisms are not so much philosophical as technical.

PROF. Y. BAR-HILLEL: They are purely technical. I said that I shall not make any philosophical criticisms, for lack of time.

DR. O. G. SELFRIDGE: A technical objection does not make ideas half-baked.

PROF. Y. BAR-HILLEL: A deductive argument, where you have first to find out what are the relevant premises, is something which many humans are not always able to carry out successfully. I do not see the slightest reason to believe that at present machines should be able to perform things that humans find trouble in doing. I do not think there could possibly exist a programme which would, given any problem, divide all facts in the universe into those which are and those which are not relevant for that problem. Developing such a programme seems to me by 10^{10} orders of magnitude more difficult than, say, the Newell-Simon problem of developing a heuristic for deduction in the propositional calculus. This cavalier way of jumping over orders of magnitude only tends to becloud the issue and throw doubt on ways of thinking for which I have a great deal of respect. By developing a powerful programme language you may have paved the way for the first step in solving problems of the kind treated in your example, but the claim of being well on the way towards their solution is a gross exaggeration. This was the major point of my objections.

DR. MCCARTHY (in reply): Prof. Bar-Hillel has correctly observed that my paper is based on unstated philosophical assumptions although what he means by “pseudo-philosophical” is unclear. Whenever we program a computer to learn from experience we build into the program a sort of epistemology. It might be argued that this epistemology should be made explicit before one writes the programme, but epistemology is in a foggier state than computer programming even in the present half-baked state of the latter. I hope that once we have succeeded in making computer programs reason about the world, we will be able to reformulate epistemology as a branch of applied mathematics no more mysterious or controversial than physics.

On re-reading my paper I can't see how Prof. Bar-Hillel could see in it a proposal to specify a computer program carelessly. Since other people have proposed this as a device for achieving "creativity", I can only conclude that he has some other paper in mind.

In his criticism of my use of the symbol "*at*", Prof. Bar-Hillel seems to have misunderstood the intent of the example. First of all, I was not trying to formalize the sentence form, A is at B, as it is used in English. "*at*" merely was intended to serve as a convenient mnemonic for the relation between a place and a sub-place. Second, I was not proposing a practical problem for the program to solve but rather an example intended to allow us to think about the kinds of reasoning involved and how a machine may be made to perform them.

Prof. Bar-Hillel's major point concerns my statement that the premises listed could be assumed to be in memory. The intention of this statement is to explain why I have not included formalizations of statements like, "it is possible to drive from my home to the airport" among my premises. If there were n known places in the county there would be

$$\frac{n(n-1)}{2}$$

such sentences and, since we are quite sure that we do not have each of them in our memories, it would be cheating to allow the machine to start with them.

The rest of Prof. Bar-Hillel's criticisms concern ways in which the model mentioned does not reflect the real world; I have already explained that this was not my intention. He is certainly right that the complexity of the model will have to be increased for it to deal with practical problems. What we disagree on is my contention that the conceptual difficulties arise at the present level of complexity and that solving them will allow us to increase the complexity of the model easily.

With regard to the discussion between Prof. Bar-Hillel and Oliver Selfridge — the logic is intended to be faultless although its premises cannot be guaranteed. The intended conclusion is "$do(go(desk, car, walking))$"—not, of course, "$at(I, airport)$". The model oversimplifies but is not intended to oversimplify to the extent of allowing one to deduce one's way to the airport.