New Generation Computing, 8 (1990) 91-93
OHMSHA, LTD. and Springer-Verlag

NEW GENERATION COMPUTING

Preface

English as a Logic Programming Language

Robert KOWALSKI
Department of Computing,
Imperial College of Science, Technology and Medicine,
180 Queen's Gate, London SW7 2BZ, England.

Ordinary English is generally ambiguous, imprecise and unclear. This makes it less than ideal both for human communication and for communication with computers. To improve the use of English for human communication, English scholars have identified guidelines for good use. These guidelines help to improve clarity by reducing ambiguity, and by reducing the distance between syntactic form and semantic content.

I believe that English formulated in accordance with the guidelines for good use can also serve as a yardstick for evaluating computer languages and knowledge representation formalisms. The language of public notices is especially useful in this respect. It tends to combine the precision of legal language with the clarity and coherence needed to be understood efficiently by ordinary people.

Consider, for example, the English of the following notice posted in the carriages of the London underground.

Emergencies

Press the alarm signal button
to alert the driver.

The driver will stop immediately
if any part of the train is in a station.

If not, the train will continue to the next station,
where help can more easily be given.

There is a £50 penalty
for improper use.

The language of the notice is close to a logic program. The first sentence, for example, which is written in procedural form can be rewritten in the declarative form we normally use for logic programs.

The driver is alerted
if you press the alarm signal button.

Because of their declarative syntax, logic programs are often considered to be non-procedural. This is misleading. The procedural interpretation of logic programs means that they can be understood and written either declaratively or procedurally. The first sentence of the underground notice shows the importance of allowing logic programs to be expressed in procedural syntax.

The second sentence illustrates an important difference between human language and computer language. Human language, even when it is unambiguous, leaves assumptions and conclusions implicit, whereas computer language forces them to be explicit. Expressed in logic programming form, the second sentence has an extra condition which is missing, but intended, in the English.

The driver stops the train immediately
if the driver is alerted
and any part of the train is in a station.

Such extra precision is necessary in computer languages, and sometimes desirable even in human languages.

The third sentence contains an explicit reference to the condition of the previous sentence. It also contains an extra phrase

where help can more easily be given.

which is strictly unnecessary, but relates the program to the ultimate goal. Assuming that the purpose of the train continuing to the next station is so that it will stop there, the third sentence can be paraphrased by two clauses of a logic program.

The driver stops the train at the next station
if the driver is alerted
and not any part of the train is in a station.

Help can be given in an emergency
if the driver stops the train in a station.

Of course, it goes without saying (in English at least) that

The driver stops the train in a station
if any part of the train is in a station
and the driver stops the train immediately.

The driver stops the train in a station
if not any part of the train is in a station
and the driver stops the train at the next station.

Such assumptions, or ones which are equivalent, have to be represented explicit-

ly, or accounted for in some other way, in logic or in any other computer language. Notice that, in order to avoid the complications of temporal reasoning, I have expressed all verbs in the logic program in the present tense.

The last sentence is for the connoisseur of logical niceties. It illustrates the kind of meaning that would normally be formalized in a specialized deontic logic of permissions, obligations, and prohibitions. But it too can be expressed in standard logic programming form.

> There is a £50 penalty
> if you use the alarm signal button improperly.

The same meaning can also be expressed (ironically, perhaps) in procedural logic programming form.

> To get a £50 penalty
> use the alarm signal button improperly.

As mentioned before, both the declarative and the procedural representations are good logic programming form.

I believe that the London underground notice exemplifies that logic programs compare well with programs written in English to be executed by people. I believe that more detailed study of good English will indicate ways to improve logic programming as a basis for developing more human-oriented computer languages of the future.