Knowledge-Based Systems, Viewpoints and the World Wide Web

I. Finch

Connect, University of Liverpool,
Liverpool, L69 3GL, United Kingdom
ian@connect.org.uk

Abstract

The World Wide Web is becoming increasingly popular as a medium for disseminating advice or help to users of products or services. Some of these advice systems are based around knowledge-based systems and as such, can exploit many of the advantages of this technology. However, a wide range of users come into contact with a Web site, bringing with them differing needs. In order to tackle this issue, the *viewpoints* mechanism, devised by the author to address such problems in more conventional knowledge-based systems, can be employed. This mechanism allows different parts of a single underlying knowledge base to be utilised, presenting a different view of the knowledge base to different groups of users. By combining the viewpoints mechanism with the Web, a single knowledge base can provide advice in a number of ways for different classes of user. For example, a council can provide advice directly to users through the Web, via telephone operatives on a help line and to claim adjudicators, with each interaction tailored to that user's needs.

Introduction

The World Wide Web (or Web) is becoming increasingly popular as a medium for disseminating advice or help to users of products or services. Some of these advice systems are based around knowledge-based systems and as such, can exploit many of the advantages of this technology. However, a wide range of users come into contact with a Web site, bringing with them differing needs. Whilst this has always been an issue with knowledge-based systems (De Carolis et al. 1996) the much larger user population that have access through the Web mean that it is now a far more significant issue which needs addressing.

In order to tackle this issue, the *viewpoints* mechanism, devised by the author to address such problems in more conventional knowledge-based systems, can be employed. The viewpoints mechanism (Finch 1993) allows different parts of a single underlying knowledge base to be utilised, presenting a different view of the knowledge base to different groups of users. Indeed, different viewpoints can be used for reasoning as well as for the presentation of information. For example, a system can be envisaged which offers advice to benefit claimants on how to claim welfare payments. The same system could also be used by people adjudicating those claims, but using a different viewpoint. It is enough for a mother to know that they can claim benefits for their children. The adjudicator, on the other hand, needs to refine this requirement to include some proof that the mother actually has children. The viewpoints mechanism allows such a situation to be represented in a single knowledge base.

Thus, by combining the viewpoints mechanism with the Web, a single knowledge base can provide advice in a number of ways for different classes of user. A council can provide advice directly to users through the Web, via telephone operatives on a help line and to claim adjudicators, with each interaction tailored to that user's needs.

The next section presents a brief discussion of the IMVEX rule-based shell, which supports the viewpoints mechanism. This is necessary to understand later examples. Further sections will then discuss the concept of viewpoints and their application to the Web.

The IMVEX Shell

The IMVEX shell is coded as a suite of Perl modules; (Wall, Christiansen, and Schwartz 1996). It is derived from an earlier version, written in Prolog and described in (Finch 1994).

IMVEX uses a backward-chaining inference engine. Starting with a top-level goal it tries to find a value using the following strategy:

- Check whether a value has already been deduced for the goal
- Try to deduce a value for the goal from the rule base (which will cause sub-goals to be sought, using the same strategy)
- Ask the user

The rules are simply expressed as a label, a conjunction of conditions and a list of the resulting conse-

quences. For example, the following rule (whose label is '4') states that any man, under the age of 65, who is healthy, is not entitled to parish relief:

```
4: age < 65 & man & healthy => relief'no
```

As can be seen from this rule, IMVEX conditions and consequences can either consist of a symbol and a binding (such as `relief'no`) or a numeric comparison (such as `age < 65`). The conditions which consist solely of a symbol have an implicit binding of 'yes' (so `man` is really `man'yes`). So, this rule will succeed if the numeric value bound to 'age' is less than 65, the value 'yes' is bound to 'man' and the value 'yes' is bound to 'healthy'. If the rule succeeds, this results in the value 'no' being bound to 'relief'.

This rule may be presented to the user at some point, in which case it is displayed in the following way:

```
Rule 4 states that:

   [1] IF age'<65
   [2] AND man'yes
   [3] AND healthy'yes
   [4] THEN relief'no
```

This is not very friendly, so IMVEX allows text to be easily associated with symbols and their values. For example:

```
age'<65: you are under the age of 65
man'yes: you are a man
healthy'yes: you are healthy
relief'no:
   you will not be granted parish relief
```

These are then used wherever text is presented to the user, for example the above rule becomes:

```
Rule 4 states that:

   [1] IF you are under the age of 65
   [2] AND you are a man
   [3] AND you are healthy
   [4] THEN you will not be granted parish
         relief
```

Similarly, text can be associated with questions, so that they can be understood by the user:

```
age'question: How old are you
```

IMVEX also has various standard features common to most rule-based systems, notably it can answer the standard 'why' and 'how' questions.

The important thing to note about the current implementation of IMVEX is its modular nature. It consists of one module which can parse the rule base and a second module which performs the reasoning as described above (and answers questions as a side effect). What it does not do is read in a rule base from anywhere, display any output or take any input. An interface module must be written for each desired environment which performs these tasks.

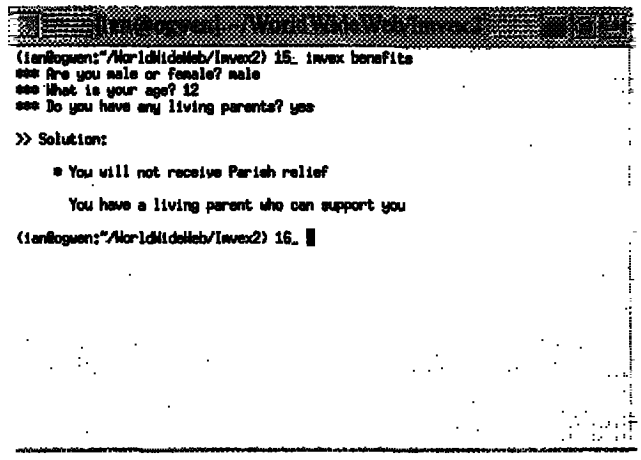

Figure 1: Accessing IMVEX through the Command Line

Currently, there are two interface modules for IMVEX: a simple text-based interface and a Web interface. Both interfaces currently read in their rule base from a plain text file. The text-based interface displays its output to the user's screen and accepts answers to questions from the keyboard (figure 1). The Web-based interface produces its output as Web pages and allows the user to click on the answers to their questions (figure 2).

Other interfaces can be envisaged, with the most obvious perhaps being an e-mail interface. Indeed, earlier work by the author used e-mail as a communication medium between knowledge-based systems (Finch et al. 1997) although this relied on a purpose-built e-mail user agent. Another interface could be the same as the Web-based interface described above, but instead of reading the rule base from a local file on the Web server, it could be loaded from any other Web server on the Internet.

Another useful interface module would not immediately display questions for the user to answer, but would first check a database to see whether the answer to that question has been stored for some reason. For example, imagine a helpline operator at a council, attempting to answer a caller's question about housing repairs. The system need not ask the operator about the state of repair of the caller's house, since it can get that information directly from the council's works database.

This, then, gives sufficient information about IMVEX and its workings to understand the remainder of the examples. Before proceeding, however, it is worth introducing an example scenario which allows a more concrete discussion of the principles involved.

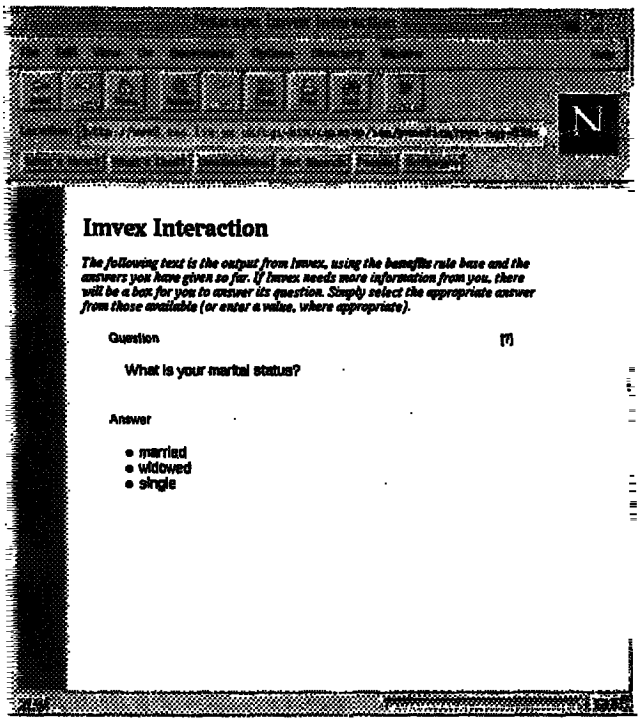

Figure 2: Accessing IMVEX through the Web

An Example Scenario

The domain used as an example in this paper is taken from (Bench-Capon 1991), and was chosen because it is small enough to understand in its entirety whilst exhibiting the characteristics of many real-world problems. The domain is the parish relief law of a fictional parish in Elizabethan England. The legislation is that:

> *Any person shall be provided with parish relief if he apply to a beadle of the parish, and in the opinion of that beadle he is deserving of such relief.*

Furthermore, there are two test cases which have extended the parish law. Firstly, it was held that John Goodbody was not deserving of relief because he was a sturdy vagabond who could make his own shift (that is, he could provide for himself). In the second case, relief was granted to Mistress Quickly, a widow, when the judge held that of all people widows and orphans were the most deserving of relief. This leads to the following categories of people who are not deserving of relief:

- Men of good health under the age of 65 (from the John Goodbody case).
- Married women (who can be supported by their husbands).
- Single women of good health under the age of 60 (who can make their own shift or marry).
- Anyone under the age of 14 with a living parent (who will be supported by their parents).

Similarly, the following categories of people will be granted relief:

- Widows (from the Quickly case)
- Anyone who is unsturdy (since they cannot make their own shift)
- Children under 14 with no living parents (a definition of orphan)
- Men over the age of 65
- Women over the age of 60

Viewpoints

The viewpoint mechanism allows a single knowledge base to provide different perspectives to different users, emphasising some aspects and hiding others. It has been discussed at length in other articles, and the interested reader is directed to (Finch 1993) and (Finch 1994) for further information. It is sufficient to discuss here the application of viewpoints to the example domain, and their implementation within IMVEX.

Consider two participants in an application for parish relief: the claimant and the beadle (or his modern day counterpart, the claim adjudicator). The claimant makes an application for parish relief and the adjudicator decides whether that claim is valid. Viewpoints allow both participants to use the same rule base, but to interact with it and get information in a way suited to their needs.

The simplest example of this is the presentation of information. A claimant will want information and questions targetted at themselves, whilst the adjudicator will want them to refer to the claimant. This can be easily accomplished by associating viewpoints with the translations, for example:

```
relief'yes/claimant:
   you will receive parish relief
relief'yes/adjudicator:
   the claimant will receive parish relief
```

There can still be a 'neutrally phrased' question in the rule base, not associated with any viewpoint:

```
relief'yes: parish relief will be granted
```

A viewpoint is said to be 'active' or 'inactive'. If the claimant viewpoint is active, the first form will be used, whereas the second form will be used if the adjudicator viewpoint is active. If no viewpoint is active, the form which is not associated with a viewpoint will be used. If both viewpoints are active, the earliest form in the rule base will be used.

Although this is a useful facility, it is only a minor aspect of the viewpoints mechanism, affecting only the superficial presentation of information. The major advantage of the viewpoints mechanism is the ability to

associate viewpoints with parts of the rulebase, allowing alternative chains of reasoning to be formed, depending on the active viewpoint. Consider the following rule which states that someone who is unhealthy should be granted parish relief:

```
9: healthy'no => relief
```

For the claimant, this is sufficient. They will be asked whether they are healthy. If they say 'no', they will be told they deserve parish relief. The adjudicator, on the other hand, needs some proof of the claimant's bad health, for example a doctor's certificate. The above rule can therefore be made part of the claimant's viewpoint. The adjudicator will have a slightly different version in their viewpoint, incorporating the need for a doctor's certificate:

```
9/claimant: healthy'no => relief
9/adjudicator:
        healthy'no & doctor_cert => relief
```

When the adjudicator is using the system, not only will they be asked whether the claimant is unhealthy, they will also be asked whether the claimant has a doctor's certificate. Thus, by using viewpoints, a single rule base can be used by both a benefit claimant and the claim adjudicator. This has obvious advantages in the maintenance and development of the rule base, since there is only one rule base to maintain. Furthermore, the models can be contradictory, as in the above example, where the claimant's rule for being granted parish relief has different conditions to the adjudicator's rule.

Scalability

This small example illustrates the principles involved, but cannot indicate how the viewpoints technique will operate on a larger scale. The issue of scalability is important in two ways: handling larger and more complex domains; and handling a larger number of viewpoints.

The first point to note is that the viewpoints mechanism effectively reduces the complexity of a domain (as it is represented in a knowledge base), since it removes parts of the knowledge base during reasoning. The search space is therefore reduced and smaller search trees can be built and traversed. However, the price for this, is a greater effort needed to create and maintain the rule base. This is only an issue when different viewpoints contain contradictory models, in which case care must be taken assigning rules to the appropriate viewpoint. There is insufficient space to present a more complex example here, but the interested reader is directed to a fuller discussion of this domain, using viewpoints to resolve several contradictions (Finch 1994) and also a discussion of a more complex domain, involving electrostatic hazards in the petroleum industry (Finch 1993).

Moving on to consider the issue of a large number of viewpoints, this is particularly relevant to the Web, since there could be hundreds or thousands of diverse users accessing a system through the Web. Although this seems a huge problem, it is important to note some points. The first is that the knowledge base (and its viewpoints) will be centered around a single domain, so the users of the system can never be too disparate. An example may be a system to offer help in repairing domestic appliances. This could cater for the home user (offering suggestions such as changing a fuse) and also for a skilled service engineer (offering detailed information about replacing the power supply). However, since 'boiling an egg' is not part of the 'repairing domestic appliances' domain, there will be no viewpoints for how to use the appliance.

The second point to note is that the viewpoints are pre-defined; a user cannot define a new viewpoint specific to themselves (at least, not yet). So, continuing with the domestic appliance theme, there might be a third category of user (somewhere between the home user and the skilled service engineer) who can perform simple repairs (say, replacing an element in an electric cooker) and is not catered for by the existing viewpoints. In the current approach, this would require the knowledge base engineers to create the viewpoint and then add whichever rules are appropriate for that viewpoint.

It is hoped that in the future, users *will* be able to create their own viewpoints 'on the fly'. The first step towards achieving this is to create micro-viewpoints from which the viewpoints can be assembled. In the domestic appliance scenario, for example, many micro-viewpoints could be created, each representing slightly more complex tasks. Rules could then be associated with each micro-viewpoint (as appropriate). Then the viewpoints could be created by listing the micro-viewpoints which make them up. The home user may only consist of the simplest micro-viewpoints whilst the skilled service engineer's viewpoint contains every micro-viewpoint. Then, if a new viewpoint is needed (like the person who can perform simple repairs), it can be assembled from the appropriate micro-viewpoints.

Conclusions

The principle of viewpoints has been discussed, together with an example of their application. The versatility of a multi-viewpoint knowledge-based system has been illustrated, particularly with regard to Web-based delivery of advice. Viewpoints allow:

- the presentation of information in different ways to different users
- the use of multiple (possibly contradictory) models of the same situation
- the system to reason differently for different users

A demonstrator system was presented which illustrates the power of this approach by providing a Web-based benefits advice service for the use of a potential benefit claimant and the claim adjudicator (with a

more complex set of rules made available). Both systems are driven by the same rule base.

The work described here is a 'proof of concept' implementation. Future work will use the approach described here in two systems. The first is for a local council and will offer advice on housing benefit, both directly to the user (through public access Internet terminals) and through a helpline operator. This will involve interaction with existing databases (as described earlier)¿ The second is for a home appliance manufacturer to provide support to both their customers and their service engineers in the field. This is a more ambitious project, since it will also include multimedia content, such as images and movie clips.

References

Bench-Capon, T.J.M. 1991. Using a Common Knowledge Base in Several Applications, In *Knowledge-Based Systems and Legal Applications*, 129–136. London, England: Academic Press.

De Carolis, B.; de Rosis, F.; Grasso, F.; Rossiello, A.; Berry, D.C.; and Gillie, T. 1996. Generating Recipient-Centered Explanations About Drug Prescription. *Artificial Intelligence in Medicine* 8(2):123–145.

Finch, I. 1993. Viewpoints — Facilitating Expert Systems for Multiple Users. In *Database and Expert Systems Applications, 4th International Conference Dexa '93*, 401–412. Vienna, Austria: Springer-Verlag.

Finch, I. 1994. A Mechanism for Creating Knowledge Bases Supporting Contradictory Models. In *Research and Development in Expert Systems, Proceedings of Expert Systems 1994*, 188–213. Oxford, England: Information Press Ltd.

Finch, I.; Coenen, F.P.; Bench-Capon, T.J.M.; and Shave, M.J.R. 1997. Coordinating Human and Software Agents through Electronic Mail. In *Cooperative Knowledge Processing*, 64-76. London, England: Springer-Verlag.

Wall, L.; Christiansen T.; and Schwartz R.L. 1996. *Programming Perl.* Sebastopol, California: O'Reilly and Associates.