Users

How can users provide knowledge when

- they don’t know the internals of the system
- they aren’t experts in the domain
- they don’t know what information is relevant
- they don’t know the syntax of the system
- but they have essential information about the particular case of interest?

Querying the User

- The system can determine what information is relevant and ask the user for the particular information.
- A top-down derivation can determine what information is relevant. There are three types of goals:
  - Goals for which the user isn't expected to know the answer, so the system never asks.
  - Goals for which the user should know the answer, and for which they have not already provided an answer.
  - Goals for which the user has already provided an answer.

Yes/No questions

- The simplest form of a question is a ground query.
- Ground queries require an answer of “yes” or “no”.
- The user is only asked a question if
  - the question is askable, and
  - the user hasn’t previously answered the question.
- When the user has answered a question, the answer needs to be recorded.

Ask-the-user meta-interpreter

% $aprove(G)$ is true if G is a logical consequence of the
% base-level KB and yes/no answers provided by the user.

$$aprove(true).$$

$$aprove((A \,\&\, B)) \leftarrow aprove(A) \wedge aprove(B).$$

$$aprove(H) \leftarrow askable(H) \wedge answered(H, yes).$$

$$aprove(H) \leftarrow$$
$$\quad askable(H) \wedge unanswered(H) \wedge ask(H, Ans) \wedge$$
$$\quad record(answered(H, Ans)) \wedge Ans = yes.$$

$$aprove(H) \leftarrow (H \Leftarrow B) \wedge aprove(B).$$

Functional Relations

- You probably don't want to ask $?age(fred, 0)$, $?age(fred, 1)$, $?age(fred, 2)$, $\ldots$
- You probably want to ask for Fred's age once, and succeed for queries for that age and fail for other queries.
- This exploits the fact that *age* is a functional relation.
- Relation $r(X, Y)$ is functional if, for every X there exists a unique Y such that $r(X, Y)$ is true.

Getting information from a user

- The user may not know the vocabulary that is expected by the knowledge engineer.
- Either:
  - The system designer provides a menu of items from which the user has to select the best fit.
  - The user can provide free-form answers. The system needs a large dictionary to map the responses into the internal forms expected by the system.

More General Questions

Example: For the subgoal $p(a, X, f(Z))$ the user can be asked:

for which X, Z is $p(a, X, f(Z))$ true?

- Should users be expected to give all instances which are true, or should they give the instances one at a time, with the system prompting for new instances?

Example: For which S, C is $enrolled(S, C)$ true?

- Psychological issues are important.

Reasking Questions

When should the system repeat or not ask a question?

Example:

Query	Ask?	Response
$?p(X)$	yes	$p(f(Z))$
$?p(f(c))$	no	
$?p(a)$	yes	yes
$?p(X)$	yes	no
$?p(c)$	no	

Don't ask a question that is more specific than a query to which either a positive answer has already been given or the user has replied *no*.

Delaying Asking the User

- Should the system ask the question as soon as it's encountered, or should it delay the goal until more variables are bound?

- Example consider query $?p(X)$ & $q(X)$, where $p(X)$ is askable.
  - If $p(X)$ succeeds for many instances of X and $q(X)$ succeeds for few (or no) instances of X it's better to delay asking $p(X)$.
  - If $p(X)$ succeeds for few instances of X and $q(X)$ succeeds for many instances of X, don't delay.